Modelling of a helicopter-based rocket launching system

A Landman

orcid.org 0000-0002-5226-3588

Thesis submitted in fulfilment of the requirements for the degree *Doctor of Philosophy in Electrical and Electronic Engineering* at the North-West University

Promoter: Dr JJ Krüger

Graduation: October 2018
Student number: 11557311

Acknowledgements

I wish to acknowledge the contributions made to this study by my colleagues of the Rooivalk Team at Denel Aeronautics, especially Mr Andries Janse van Rensburg who on many occasions found the time and energy to discuss with me various problems that I encountered on this journey of discovery and innovation.

I am also very much indebted to my promotor, Dr Johann Kruger who had the knowledge, insight and enthusiasm to lead me in this endeveour.

Finally, a very warm thank you to my wife Marieta who encouraged me and made it possible for me to perform this study at her expense in time during many years of part time study.

Aan Marieta
Jy is my krag en inspirasie

Abstract

This study describes a conceptual design of a weapon system that employs a Model Predictive Control Loop in conjunction with an Array Processor to deliver standard Wrap Around Fin rockets with high precision from a helicopter. One of the requirements of such a system is real time estimation of the main rotor downwash. The design thread created by this requirement is followed by this study, starting with the operational requirement at system level and ending with porting of code into Field Programmable Gate Arrays at component level.

To prove that real time estimation of the main rotor downwash of a Helicopter is feasible a conceptual design of a Precision Rocket Launcher with integral Array Processor based on Field Programmable Gate Arrays is developed. An algorithm compatable with the Array Processor and suitable for solving a taylored version of the Navier Stokes equation in real time is developed. Two versions of this algorithm called the Concurrent Model and Pipeline Model are subsequently prepared for porting to Field Programmable Gate Arrays.

The Concurrent Model is a direct implementation of the downwash algorithm in 20 bit fixed point notation. Due to its simplicity this model is capable of solving the taylored version of the Navier Stokes equation at an iteration rate of 63,4 kHz when ported to Intel device GX2800 of the STRATIX 10 family. The Concurrent Model is shown to have an unfavourable Multiplier to Fabric ratio which results in such a large component count that packaging and cooling them within the confines of the Precision Rocket Launcher is unfeasible.

The Pipeline Model implements the downwash algorithm with seven pseudo-processors arranged in a pipeline configuration. It is implemented in single precision floating point notation and is capable of solving the Taylored Navier Stokes equation at a rate of 10 kHz when ported to Intel device GX2800 of the STRATIX 10 family. The Pipeline Model is shown to have a favourable Multiplier to Fabric ratio which results in a reasonable component count, making packaging and cooling them within the confines of the Precision Rocket Launcher feasible.

The Pipeline Model yields a computational domain of 525x525x525 cells if it is implemented on an Array Processor based on Intel device GX2800 of the STRATIX 10 family. Throughput of this Array Processor is 0,601 petaflops (single precision) with RAM data flow rate of 0,261 petabytes/sec.

The downwash models, as developed in this study, form standard building blocks which can also be used in other real time applications. The models use a relatively simple interface with any combination of system elements such as rotors, wings, fuselage, ground, gas sources/sinks and other obstructions which makes real time calculation of airflow in many configurations and conditions possible.

Keywords: Massive parallelism, real time, rotor downwash, array processor, Field Programmable Gate Array, helicopter, airborne application, pseudo-processor, Navier Stokes equation, cost to benefit ratio, flow pattern, Precision Rocket Launcher, ballistics model, Model Predictive Control Loop, rocket trajectory, Data Flow Block Diagram, interface, system segment, rotor, wing.

Opsomming

Hierdie studie beskryf ‘n konsepsuele ontwerp van ‘n wapenstelsel wat ‘n Voorspellingsmodelbeheerlus met ‘n Matrysprosesseerder kombineer om standaard Omvouvlerkvuurpyle met presisie vanaf ‘n helikopter af te lewer. Een van die vereistes van so ‘n stelsel is intydse skatting van die rotorwas. Die ontwerpsroete wat as gevolg van hierdie vereiste ontstaan, word deur hierdie studie gevolg, vanaf die operasionele behoefte op stelselvlak tot by the aflaai van kode in Veldprogrammeerbare Hekmatryse op komponentvlak.

Ten einde aan te toon dat intydse berekening van die hoofrotorstroom van ‘n Helikopter moontlik is, word ‘n konsepsuele ontwerp van ‘n Presisie Vuurpyllanseerder met integrale Matrysprosesseerder, gebaseer op Veldprogrammeerbare Hekmatryse, ontwikkel. ‘n Algoritme wat geskik is vir implimentering op die Matrysprosesseerder en gebruik kan word om ‘n aangepaste Navier Stokes vergelyking op te los, word afgelei. Twee weergawes van hierdie algoritme genaamd die Konkurrente-model en Pyplynmodel word ontwikkel vir aflaai in Veldprogrammeerbare Hekmatryse.

Die Concurrent Model is ‘n direkte implimentering van die aangepaste Navier Stokes vergelyking. As gevolg van sy eenvoud kan hierdie model die rotorwas skat teen ‘n iterasietempo van 63,4 kHz indien dit met komponent GX2800 van die Stratix 10 familie familie geimplimenteer word. Daar word aangetoon dat die Concurrent Model ‘n ongunstige vermenigvuldiger tot matrysmateriaal verhouding het. Dit gee aanleiding tot ‘n groot aantal komponente wat hul verpakking en verkoeling in die Presisie Vuurpyllanseerder onwerkbaar maak.

Die Pipeline Model implimenteer die aangepaste Navier Stokes vergelyking met sewe pseudo-prosesseerders in ‘n pyplyn konfigurasie. Hierdie model is in enkelpresisie glypuntnotasie geimplimenteer en kan die rotorwas skat teen ‘n tempo van 10 kHz indien dit met komponent GX2800 van die Stratix 10 familie familie geimplimenteer word. Daar word aangetoon dat die Pipeline Model ‘n gunstige vermenigvuldiger tot matrysmateriaal verhouding het. Dit gee aanleiding tot minder komponente wat hul verpakking en verkoeling in die Presisie Vuurpyllanseerder moontlik maak.

Die Pipeline Model lewer ‘n berekeningsvolume van 525x525x525 selle indien dit geimplimenteer word met ‘n Matrysprosesserder wat op komponent GX2800 van die Stratix 10 familie gebaseer is. Deurset van sodanige Matrysprosesseerder is 0,601 petaflops (enkelpresisie) met RAM datavloeitempo van 0,261 petabytes/sec.

Die rotorwasmodelle soos in hierdie studie ontwikkel is vorm standaard boublokke wat ook in ander reëletyd toepassings gebruik kan word. Die modelle gebruik ‘n relatief eenvoudige intervlak met enige kombinasie van stelselelemente soos rotors, vlerke, romp, grond, gasbronne/sinke en ander obstruksies wat intydse berekening van lugvloei in vele konfigurasies en kondisies moontlik maak.

Sleutelwoorde: Massiewe parallelisme, reële tyd, rotorwas, matrys prosesseerder, Veldprogrammeerbare Hekmatrys, helikopter, vliegtuig toepassing, pseudo-prosesseerder, Navier Stokes vergelyking, kostevoordeelverhouding, vloeipatroon, Presisie Vuurpyllanseerder, ballistiese model, Voorspellingsmodelbeheerlus, vuurpyltrajek, datavloeiblokdiagram, intervlak, stelselement, rotor, vlerk.

Table of Contents

List of Figures

List of Tables

Nomenclature

l	:	Length of heat sink [m]
$\dot{m}_{mr}$	:	Mass flow rate through main rotor [kg/s]
$\dot{m}_e$	:	Mass flow rate through nozzle [kg/s]
$\dot{m}_{ec}$	:	Mass flow rate through choked nozzle [kg/s]
A_e	:	Surface area of nozzle exit [m^2]
A_o	:	Frontal cross-sectional surface area of rocket [m^2]
A_r	:	Lateral cross-sectional surface area of rocket [m^2]
A_δ	:	Constant of proportionality between average wind speed normal to the Wrap Around Fin surface and spin rate of the rocket
C_{Di}	:	Induced drag coefficient of rocket
C_{Do}	:	Zero-yaw drag coefficient of rocket
C_{lp}	:	Perturbation angle of rocket centre line from the relative air velocity $\boldsymbol{V}_a$ [rad]
$C_{l\delta}$	:	Roll Moment Coefficient induced by canting of a Wrap-Around Fin
C_{mp}	:	Roll Moment Coefficient caused by perturbation of the FZ90 Rocket centre line
C_{ms}	:	Roll Moment Coefficient caused by spin of the rocket
D_r	:	Reference diameter (0,07 meter) of MK-66 Rocket Motor [m]
$IDC_{ZXY}(\phi_r, \theta_r, \psi_r)$	:	Inverse ZXY direction cosine matrix resulting from yawing, pitching and rolling the rocket through the angles ϕ_r,θ_r and ψ_r in the same order
M_r	:	Relative wind speed from perspective of a rocket expressed as ratio of local speed of sound [Mach no]
M_{waf}	:	Overall torque generated by Wrap Around Fin along the **Y**r Rocket Axis [Nm]
P_∞		Ambient pressure in vicinity of nozzle [N/m^2]
P_e		Pressure at nozzle exit [N/m^2]
V_{ary}	:	Component of $\boldsymbol{V}_{ar}$ along the **Y**r Rocket Axis [m/s]
V_e		Gas exit velocity with reference to rocket [m/s]
f_{nt}	:	Nyquist frequency of the Taylored Navier Stokes Solver [Hz]
ur_x	:	Component of unit vector **ur** along $\boldsymbol{X}_i$ axis [m/s]
ur_y	:	Component of unit vector **ur** along $\boldsymbol{Y}_i$ axis [m/s]
ur_z	:	Component of unit vector **ur** along $\boldsymbol{Z}_i$ axis [m/s]
$\boldsymbol{F}_D$	:	Total drag force acting on rocket in Inertial Axes [N]
$\boldsymbol{V}_a$	:	Relative air velocity in Inertial Axes from perspective of rocket [m/s]
$\boldsymbol{V}_r$	:	Velocity of rocket in Inertial Axes [m/s]
$\boldsymbol{V}_w$	:	Wind velocity in Inertial Axes [m/s]

$\boldsymbol{X}_n$	:	Value of **X** during current iteration step
$\boldsymbol{X}_{n+1}$	:	Value of **X** during next iteration step
$\boldsymbol{X}_{n-1}$	:	Value of **X** during previous iteration step
δ_l	:	Cant angle of Wrap Around Fin [rad]
θ_r	:	Angle of rotation around $\boldsymbol{X}_i'$ axis to rotate Inertial Axes into Rocket Axes [rad]
ρ_r	:	Density of air in vicinity of rocket [kg/m^3]
ψ_r	:	Angle of rotation around $\boldsymbol{Y}_i''$ axis to rotate Inertial Axes into Rocket Axes [rad]
$\frac{\partial \boldsymbol{X}}{\partial t}$	:	Partial derivative of **X** with respect to time during current iteration step [m/s]
ϕ_r	:	Angle of rotation around $\boldsymbol{Z}_i$ axis to rotate Inertial Axes into Rocket Axes [rad]
μ	:	Viscosity of ambient fluid [Ns/m^2]
μ_a	:	Viscosity of air at air temperature of +25°C [Ns/m^2]
A	:	Surface area of equal-sided test volume face [m^2]
A	:	5x5 matrix dependant only on the current value of **X**
A	:	Surface area of body from which convection occurs [m^2]
A, B, C	:	Coefficients of second-order Taylor series that fits the values of variable *s* in three adjacent test volumes in the Flight Box
A_{mr}	:	Overall surface area of the main rotor disk [m^2]
A_p	:	Effective pumping surface area of the main rotor disk [m^2]
B	:	1x5 matrix dependant on the second partial derivatives of **X** in spatial domain
C	:	Constant to be determined through empirical means
C_a	:	Specific heat capacity of air at air temperature +25°C [Joule/kg.°C]
Cp	:	Specific heat at constant pressure of ambient fluid [Joule/kg.°C]
CS_1, CS_2, CS_3, CS_4	:	Set of coefficients used in Anti-Aliasing Spatial Filter that must be chosen to yield the required frequency response in the spatial domain
CT_1, CT_2, CT_3	:	Set of coefficients for Anti-Aliasing Temporal Filter that must be chosen to yield the required frequency response of TNS Solver in the time domain
D_{mr}	:	Diameter of the rain rotor [m^2]
G_r	:	Grashof group
h	:	Heat transfer coefficient of convection layer
$h'_{x,y,z,t}$	:	Spatially filtered version of variable h at point [x,y,z,t]
$h_{x,y,z,t}$	:	Output of Temporal Anti-aliasing Filter at point [x,y,z,t] using signal y as input
I_{mc}	:	Total number of bits exchanged between cores of **Pipeline Model** embedded in Planar Array Processor during one Major Cycle [bits]

I_{tot}	:	Total I/O data flow rate of **Pipeline Model** embedded in Planar Array Processor [bits/second]
k	:	Thermal conductivity of ambient fluid [Watt/m°K]
k_p	:	Ratio of effective pumping surface area of the main rotor disk to overall surface area of the main rotor disk
L	:	Characteristic dimension of object [m]
M_{ac}	:	All-up mass of the aircraft [kg]
m_s	:	Mass of air in test volume [kg]
N_u	:	Nusselt group
O_{mc}	:	Total number of single precision floating point operations performed by **Pipeline Model** embedded in Planar Array Processor during one Major Cycle [flops/sec]
O_{tot}	:	Total throughput of **Pipeline Model** embedded in the Planar Array Processor [flops/sec]
p	:	Pressure of air in test volume [N/m^2]
P_{10}	:	Air pressure above the main rotor disk [N/m^2]
P_{20}	:	Air pressure below the main rotor disk [N/m^2]
P_r	:	Prandtl group
Q	:	Overall heat transferred through convection layer [Watt]
R	:	Specific gas constant for air [J/kg°K]
R_e	:	Reynolds group
R_{mc}	:	Total number of bits exchanged with RAM of **Pipeline Model** embedded in Planar Array Processor during one Major Cycle [bits]
s	:	Dummy variable used to derive equation for calculating spatial derivatives of Flight Box
T	:	Temperature of air in test volume [°K]
T_1	:	Temperature of body from which convection occurs [°K]
T_2	:	Temperature of ambient fluid into which convection occurs [°K]
T_a	:	Air temperature [°K]
T_{ao}	:	Air temperature at ground level [°K]
u	:	Velocity of ambient fluid relative to body [m/sec]
ur	:	Unit vector along the rocket centre line
V_s	:	Volume of test volume [m^3]
$\boldsymbol{V}_{worb}$	:	Velocity in Rocket Axes of wind at a given point on surface of a Wrap-Around Fin with spin-induced velocity distribution [m/sec]
$\boldsymbol{V}_{wp}$	:	Component of wind velocity in Rocket Axes normal to the surface of a Wrap-Around Fin with spin-induced velocity distribution [m/sec]
$W1$	:	Dimensionless constant to be determined through empirical means
$W2$	:	Dimensionless constant to be determined through empirical means
$W3$	:	Dimensionless constant to be determined through empirical means

$\boldsymbol{X}$	:	1x5 matrix representing the state variables pressure, temperature and velocity components of a given test volume [N/m^2,°K, m, m, m]
$\mathbf{X}_{mr}$, $\mathbf{Y}_{mr}$, $\mathbf{Z}_{mr}$	:	X, Y and Z axes of Main Rotor local axes system
y	:	Output of Second Order Solver at point [*x,y,z,t*]
y_a		Height above ground [m]
β	:	Coefficient of thermal expansion [m/°K]
g	:	Gravitational acceleration [m/s^2]
ΔE_f	:	Frictional energy added to air in test volume during time interval Δt [Joule]
ΔE_k	:	Kinetic energy added to air in test volume during time interval Δt [Joule]
ΔE_p	:	Pressure energy added to air in test volume during time interval Δt [Joule]
ΔE_t	:	Thermal energy added to air in test volume during time interval Δt [Joule]
Δm_s	:	Change in mass of air in test volume during time interval Δt [kg]
ΔT	:	Temperature difference between body and ambient fluid [°K]
Δx, Δy, Δz	:	Dimensions of test volume [m]
Δx, Δy, Δz	:	Dimensions of test volume along Xi, Yi and Zi axes [m]
ρ	:	Density of ambient fluid [kg/m^3]
ρ_s	:	Density of air in test volume [kg/m^3]
Δt	:	Time step of the Taylored Navier Stokes Solver [sec]
ω_{mr}	:	Angular rate of main rotor [rad/sec]
r	:	Radius of annular control surface [meter]
dr	:	Width of annular control surface [meter]
ΔFL_{mr}	:	Lift generated by four sections of annular control surface [Newton]
$CL_{\alpha mr}$	:	Lift coefficient of rotor blade
α_{mr}	:	Angle of attack between rotor blades and air entering rotor disk [rad]
D_b	:	Width of rotor blade [meter]
dP_h	:	Power required to cause massflow rate $\dot{m}_{cs}$ through annular control surface [kW]
$\dot{m}_{cs}$	:	Massflow rate through annular control surface [kg/sec]
P_h	:	Power required by helicopter in the hover [kW]
θ_{mr}	:	Blade pitch angle [rad]
v_1	:	Velocity of inflow [m/s]
v_2	:	Velocity of outflow [m/s]
α_{mr}	:	Angle of attack [rad]
a_o	:	Coning angle [rad]

$\boldsymbol{FW}_b$	:	Blade weight of main rotor blade [Newton]
$\boldsymbol{FC}_b$	:	Centrifical force acting on main rotor blade [Newton]
$\boldsymbol{FL}_b$	:	Lift generated by main rotor blade [Newton]
BPOA	:	Blade pitch offset angle [rad]
BPMA	:	Blade pitch modulation angle [rad]
ψ_{mr}	:	Main Rotor Azimuth Angle [rad]
ε	:	Cyclic Control Lever steer angle [rad]
ρ_{mr}	:	Density of air in vicinity of Main Rotor [kg/m^3]
$\boldsymbol{X}_{mr}$	:	X axis of Main Rotor Axes
$\boldsymbol{Y}_{mr}$	:	Y axis of Main Rotor Axes
$\boldsymbol{Z}_{mr}$	:	Z axis of Main Rotor Axes
dh	:	Angle subtended by control surface on Main Rotor Disk [rad]
h	:	Angular position of control surface on Main Rotor Disk [rad]
$\hat{b}$	:	Unit vector that points from root to tip of Blade1
$\hat{c}$	:	Unit vector which points from the leading edge to the trailing edge of the blade segment
$\hat{e}$	:	Unit vector that lies parallel to vector $\boldsymbol{VT}$
$\hat{g}$	:	Unit vector formed by cross product between unit vector $\hat{b}$ and unit vector $\hat{e}$.
dr_b	:	Radial width of blade segment [m]
r_b	:	Radial distance of blade segment from flapping hinge centre [m]
$\boldsymbol{VI}$	:	Velocity of air incident on blade segment [m/s]
$\boldsymbol{V}_{bs}$	:	Orbital velocity of blade segment [m/s]
$\boldsymbol{WS}$	:	Local wind velocity [m/s]
$\boldsymbol{AS}$	:	Aircraft velocity [m/s]
R_f	:	Radial distance between Drive Axis and flapping hinge centre [m]
R_i	:	Inner radius of Main Rotor Disk [m]
R_o	:	Outer radius of Main Rotor Disk [m]
$\hat{a}$	:	Unit vector that lays parallel with vector $\boldsymbol{FR}_b$
FR_x	:	Component of vector $\boldsymbol{FR}_b$ along $\boldsymbol{X}_{mr}$ axis of Main Rotor Axes [N]
FR_y	:	Component of vector $\boldsymbol{FR}_b$ along $\boldsymbol{Y}_{mr}$ axis of Main Rotor Axes [N]
FR_z	:	Component of vector $\boldsymbol{FR}_b$ along $\boldsymbol{Z}_{mr}$ axis of Main Rotor Axes [N]
VT_x	:	Component of vector $\boldsymbol{VT}_b$ along $\boldsymbol{X}_{mr}$ axis of Main Rotor Axes [m/s]
VT_y	:	Component of vector $\boldsymbol{VT}_b$ along $\boldsymbol{Y}_{mr}$ axis of Main Rotor Axes [m/s]
VT_z	:	Component of vector $\boldsymbol{VT}_b$ along $\boldsymbol{Z}_{mr}$ axis of Main Rotor Axes [m/s]

Abbreviations

AAFSS	:	Advanced Aerial Fire Supression System
ADAS	:	Any Direction Air Data Sensor
ADC	:	Analog-to-Digital Convertor
ALM	:	Adaptive Logic Module
AoA	:	Angle of Attack
AOS	:	Omni-directional Airspeed Sensor
APKWS	:	Advanced Precision Kill Weapon System
APM	:	Array Processor Module
AR	:	Aspect Ratio
ASI	:	Aircraft Store Interface
ASIC	:	Application Specific Integrated Circuit
BAE	:	British Aerospace
BGA	:	Ball Grid Array
CCIP	:	Continuously Calculated Impact Point
CCRP	:	Continiously Calculated Release Point
CEP	:	Circular Error Probable
CFD	:	Computational Fluid Dynamics
CPU	:	Central Processing Unit
CUDA	:	Compute Unified Device Architecture
DAC	:	Digital-to-Analog Convertor
DARPA	:	Defense Advanced Research Projects Agency
DFBD	:	Data Flow Block Diagram
DSP	:	Digital Signal Processor
ECS	:	Environmental Control System
EMI	:	Electro-Magnetic Interference
EMLT	:	Electro-Magnetic Launch Tube
ENIAC	:	Electronic Numerical Integrator and Computer
FCC	:	Fire Control Computer
FDM	:	Finite Difference Model
FEM	:	Finite Element Model
FET	:	Field Effect Transistor
FFP	:	Front Face Plate

FFR	:	Free Flight Rocket
FLA	:	Forward Link Assembly
FLOPS	:	Floating Point Operations
FPGA	:	Field Programmable Gate Array
GDP	:	Gross Domestic Product
GPU	:	Graphics Processor Unit
HDL	:	Hardware Development Language
HIRF	:	High Intensity Radiated Field
HMC	:	Helicopter Mission Computer
HMSD	:	Helmet Mounted Sight Display
HP	:	Host Processor
HPC	:	High Performance Computing
HPCS	:	High-Productivity Computing System
HPM	:	High Precision Model
HSEU	:	Helmet Sight Electronic Unit
HSSF	:	High Stability Space Frame
HUD	:	Heads Up Display
IDE	:	Integrated Development Environment
IED	:	Improvised Explosive Device
IGE	:	In Ground Effect
IMS	:	Integrated Management System
INS	:	Inertial Navigation System
JTAG	:	Joint Test Action Group
LE	:	Leading Edge
LIDAR	:	Light Detection And Ranging
LRU	:	Line Replaceable Unit
LTESF	:	Low Thermal Expansion Space Frame
LTL	:	Launch Tube Lining
LVDS	:	Low-voltage differential signaling
LWH	:	Launcher Wiring Harness
MANPADS	:	Man-portable air-defense system
MC	:	Major Component
MEU	:	Main Electronics Unit
MOSFET	:	Metal Oxide Field Effect Transistor

MPCL	:	Model Predictive Control Loop
MRPS	:	Main Rotor Position Sensor
NACA	:	National Advisory Committee for Aeronautics
OAS	:	Omni-directional Airspeed Sensor
OAS	:	Omni-directional Airspeed Sensor
OGE	:	Out of Ground Effect
OTR	:	Overberg Test Range
PADS	:	Precision Air Data System
PIC	:	Peripheral Interface Controller
PLL	:	Phase Locked Loop
PRB	:	Protective Roll Bar
PRL	:	Precision Rocket Launcher
PRLS	:	Precision Rocket Launching System
PSB	:	Penta Slot Bezel
PSU	:	Power Supply Unit
PTF	:	Programmable Time Fuze
PWMA	:	Pulse Width Modulation Amplifier
RAM	:	Random Access Memory
REU	:	Rocket Electronic Unit
RFP	:	Rear Face Plate
RLA	:	Rear Link Assembly
RLS	:	Rocket Launching System
RTM	:	Real Time Model
RTOS	:	Real Time Operating System
SANDF	:	South African National Defence Force
SDF	:	Six-Degree-of-Freedom
SERDES	:	Serializer/Deserializer
SFBGA	:	Super Fine Ball Grid Array
SPI	:	Serial Peripheral Interface
SRU	:	Shop Replaceable Unit
SUD	:	System Under Design
TAC	:	Two-Axis Cradle
TE	:	Trailing Edge
TFLOPS	:	Tera Floating Point Operations

TNS	:	Taylored Navier Stokes
VHDL	:	VHSIC Hardware Description Language
VHSIC	:	Very High Speed Integrated Circuit
WEAC	:	Weapons Computer
WPCB	:	Wrap-around Printed Circuit Board
WSO	:	Weapons System Officer

1 Introduction (Cycle 1, first half)

"*The enemy, employing his small forces against a vast country, can only occupy some big cities and main lines of communication and part of the plains. Thus, there are extensive areas in the territory under his occupation that he has had to leave ungarrisoned and that provide a vast arena for our guerrilla warfare*". Mao Tse-Tung.

This chapter introduces the first half of a Scientific Method Cycle that forms part of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as alternative to smart rockets. This cycle observes the challenges of asymmetric warfare, the attempts of the global defence industry to solve the problem by launching Free Flight Rockets (FFR) from helicopters as illustrated in Figure 1 and the performance of other weapon systems that may have a bearing on the problem. Suitable questions are then posed to formulate the PRLS hypothesis.

Figure 1. Early weapon system additions to the Bell UH-1 [U.S. Army, 1964]

With the PRLS hypothesis at hand suitable predictions are formulated to prove/disprove the PRLS hypothesis. These predictions are handed down for testing in Scientific Method Cycles at lower levels of this study.

Using the results of experiments conducted throughout this study a general PRLS theory is finally developed in Chapter 8 of this document, completing the second part of the overall Scientific Method Cycle.

The main contribution made to the knowledge pool in this chapter is formulation of the concept of using a Model Predictive Control Loop as a low cost alternative to Smart Rockets. Chapter 8 adresses the proof that such a system is viable.

Note: *SI units are used throughout the document unless indicated otherwise.*

Note: *Throughout this study square brackets in captions of figures are used to denote the author of the image. Square brackets in the text are used to contain an index into the list of references at the end of the document.*

1.1 The Helicopter-Rocket marriage (Make observations)

The concept of launching Free Flight Rockets (FFR) from a helicopter originated during the Algerian War (1954 - 1962) when the French fitted Sikorsky H-34 Choctaws and Alouette II helicopters with 20 mm cannons and unguided rockets that gave the mobility and firepower necessary to deal with National Liberation Front (FLN) insurgents during the Algerian War [1]. Similar devices were mounted on Bell UH-1 helicopters by the Americans during the early stages of the Vietnam war as illustrated in Figure 1.

The reason for this Helicopter-Rocket marriage is two-fold. First, the helicopter lends itself for easy and rapid deployment over a large geographic area because it can hover and land in constrained places. The helicopter can therefore be operated with minimal infrastructure close to a shifting battle area where it can penetrate very rough terrain. Second, rockets are self-launching so the mass penalty for launching them from a helicopter is small. Combined, helicopters and rockets make it possible to project a lot of power quickly and efficiently onto an enemy that spreads his resources in time and space as teached by Mao Tse-Tung [2]. This solution to one of the problems of asymmetric warfare was the basic driver for the Advanced Aerial Fire Supression System (AAFSS) requirement of the US Army, formulated in response to the problems encountered during the Vietnam war. Ultimately it led to the advent of the *AH-1 Cobra* – the world's first dedicated attack helicopter [3].

1.1.1 Body plan of the Combat Helicopter

The Cobra was based on the Bell UH-1, suitably modified to seat a Pilot and a Weapon System Officer (WSO) in a tandem arrangement and with two Stub Wings attached to the Fuselage as illustrated in Figure 2. This arrangement was chosen to maximize the external visibility of both crew members and to provide a mechanism for carrying a range of stand-off weapons including rockets and guided missiles. A 20 mm cannon was mounted in the chin of the aircraft.

Figure 2. AH-1S Cobra with typical weapons configuration [Pakistan Army, 1980]

After the Vietnam war the US Army continued development of the attack helicopter and generated their so-called Advanced Attack Helicopter (AAH) requirement that eventually culminated in the AH-64A Apache which entered service in 1983 [4].

Many other Attack Helicopters came into existence since the AH-1 Cobra and AH-64A Apache. These include aircraft such as the Mi-24 Hind, Black Hawk, Tiger, Mongoose and Rooivalk AH-2A. Important for the purpose of this thesis is that the basic body plan of all Combat Helicopters are the same - they all have Stub Wings that carry external weapons, meaning rockets are launched from underneath the Main Rotor. This combination sets the scene for a range of error sources (22 in total) that are common to all Combat Helicopters. **Overall, this thesis addresses the most important of these error sources – disturbance of the rocket trajectory due to downwash and measures for mitigating the problem.**

Figure 3. Rooivalk AH-2A with typical weapons config [Denel Aeronautics, 2010]

For the purpose of this thesis, the Rocket Launching System (RLS) that will be modelled is that of the the Rooivalk AH-2A as illustrated in Figure 3. Development of this helicopter started in 1984 with Denel Aeronautics (then Atlas Aircraft Corporation) as main contractor. The User Requirement Specification of Rooivalk AH-2A was drafted by the South African Air Force (SAAF), based on lessons learned during the Angolan Bush War (1966 - 1989).

1.1.2 The FZ90 and MK-66 rockets

For the purpose of this study the FZ90 Rocket will be assumed as comprised of the FZ90 Rocket Motor, FZ71 High Explosive Warhead and M423 Impact Fuze as used on Rooivalk AH-2A. The FZ90 Rocket family is manufactured by Forges de Zeebrugge in Belgium. It is essentially a copy of the Hydra-70 Rocket family which is manufactured by General Dynamics. Because of unavailability of data on the FZ90 Rocket, this research often used information available for the Hydra-70, instead.

Figure 4. Hydra-70 Rocket family [Defence Industry Daily, 2014]

Development of the Hydra-70 Rocket family as illustrated in Figure 4 started from the German R4M rocket which was originally developed during the Second World War [5]. The Hydra-70 Rocket went through several upgrades that eventually culminated in the MK-66 MOD4 Rocket Motor during the 1960s. Note that this version of the MK-66 Rocket is intended for helicopter operation since it contains a K_2SO_4 rod to prevent flameouts of the helicopter engines as described by *Hawley* [11]. These upgrades were either mechanical or chemical in nature. A good example is the work by *Kim* and *Walbridge* [10] to minimize the interference between rocket and downwash by using erodible vanes that are intended to spin the MK-66 Rocket up to maximum rotation rate before it leaves the rocket launcher (at which time the torque must seize to prevent further increase in rotation rate of the rocket). *Hawley* [12] also describes an erodible vane to increase the spin rate of MK-66 Rocket upon leaving the Launcher Tube as well as a Snap-on Fairing to reduce the drag coefficient of the MK-66 Rocket in order to increase range. Most of these improvements have tapered out as the limits of the associated technologies were reached. To improve the performance of the Free Flight Rocket further still now requires contributions from the electronic world such as the Advanced Precision Kill Weapon System (APKWS), Guidance Kit as described by *Wonnacott* [9] or the Precision Rocket Launching System (PRLS) as described in this study.

1.1.3 The need to avoid collateral damage

Many of the wars of the past century such as the Algerian War (1954 - 1962), Vietnam War (1959 - 1975), Afghan War (2001 - present) and Iraqi War (2003 - 2011) share a common characteristic in that they were asymmetric in nature. This tendency stems from the fact that many of these wars saw a major power poised against a much smaller opponent, forcing the smaller opponent to employ asymmetric warfare in which any conceivable tactic including those as described by Mao Tse-Tung are employed [6].

One of the tactics employed during asymmetric warfare is the use of civilians and their homes as cover and shields. This happened (amongst others) during Operation Enduring Freedom and Operation Iraqi Freedom when el Quaeda, Taliban and Iraqi forces dispersed amongst the local population, introducing the possibility of collateral damage which was politically and morally un-acceptable from the perspective of the Allied Forces [6][7][8].

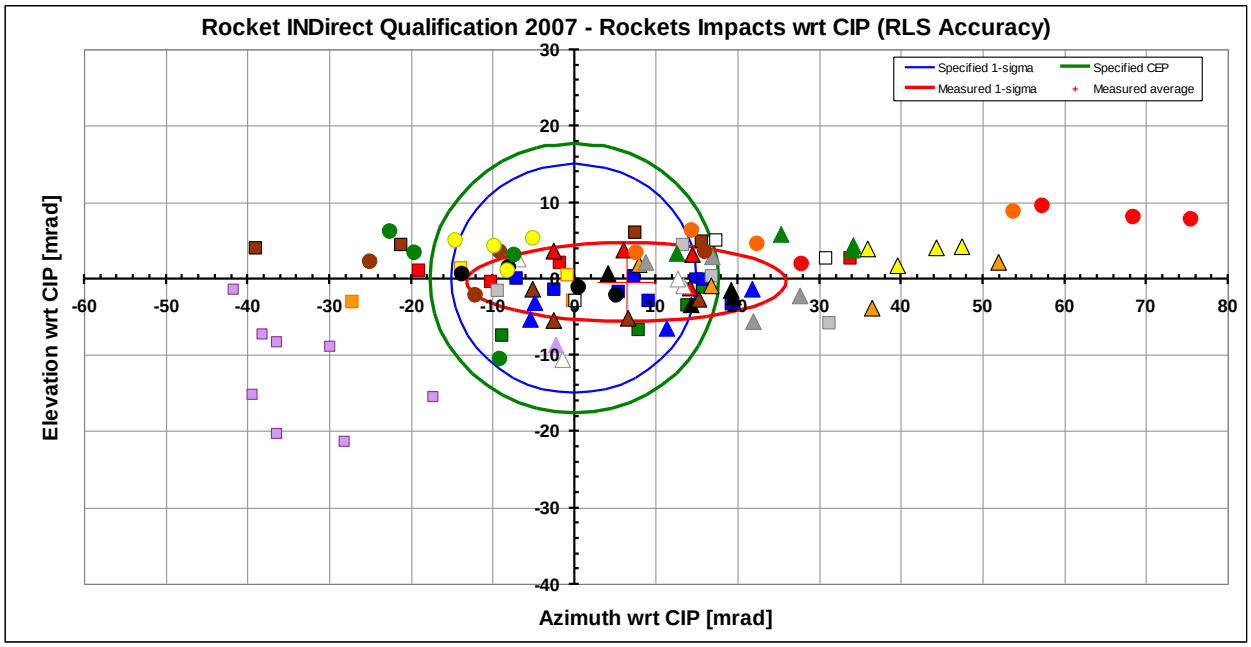


Figure 5. Summary of impact points, Rooivalk RLS qualification [Denel Aeronautics, 2007]

Unfortunately, the accuracy achieved when standard Free Flight Rockets are launched from a helicopter-based weapons platform can at best be described as that of an area weapon. Figure 5 summarizes the impact points as achieved during the qualification campaign of the Rocket Launching System (RLS) of Rooivalk AH-2A as viewed through the Main Sight of the aircraft. In this figure the different symbols represent different sorties, some of which were performed on different days. During these tests the aircraft was hovering 200 feet above ground at a range of 2,5 kilometers, so the dimensions of the ground footprint formed by these rocket impact points were about 300 meter wide by 3000 meter long.

Note: *The data represented in Figure 5 is not available in the open literature. It was obtained by the Author during his involvement with development of Rooivalk AH-2A.*

It is clear that without suitable modification the standard Helicopter-Rocket marriage is not suitable to prevent collateral damage associated with asymmetric warfare. This has been an area where contemporary electronics have already provided some mitigation of the problem by providing Smart Rockets which have very high delivery precision but are limited in destructive power [13][14][15].

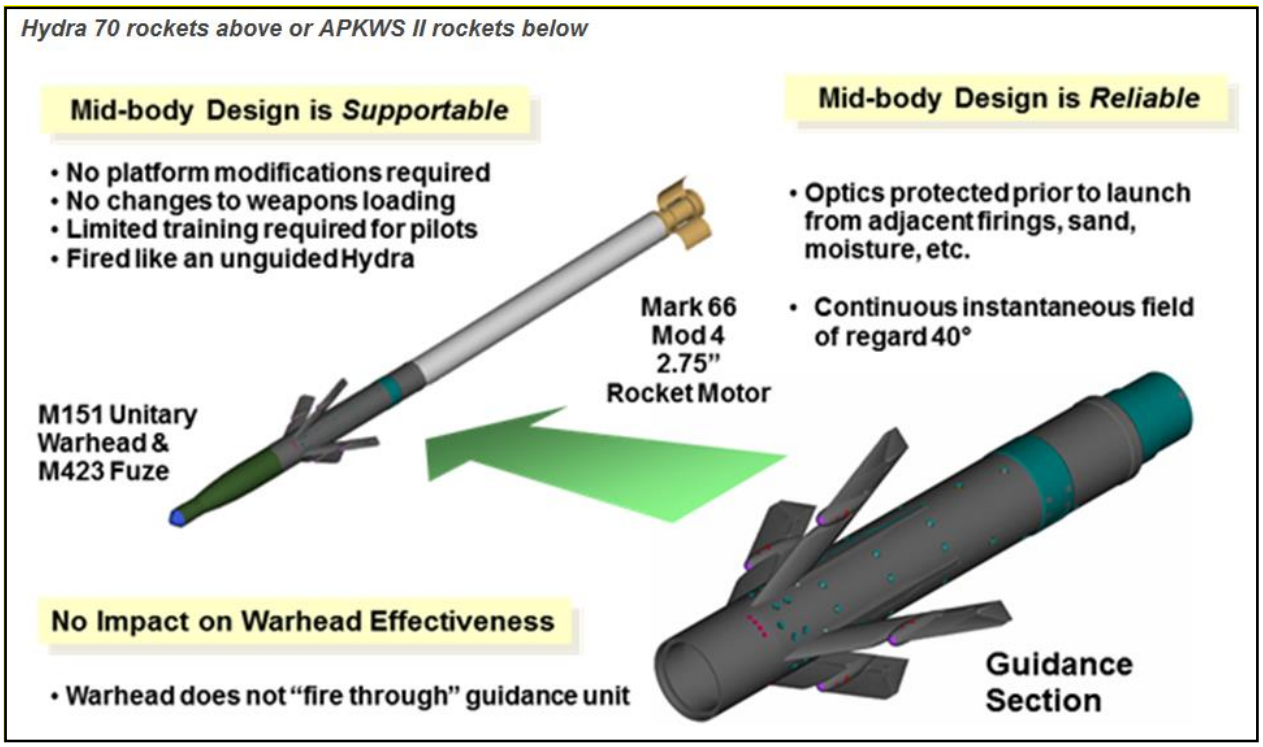


Figure 6. The APKWS Rocket Conversion [BAE Systems, 2017].

As described by *Wonnacot* [9] the need for a low cost precision weapon was recognized by the US Army in the mid 1990's. This resulted in the US Army providing initial funding for fitting the Hydra-70 Rocket with a semi-active laser seeker, resulting in the Advanced Precision Kill Weapon System (APKWS) as illustrated in Figure 6. This program was spearheaded by British Aerospace and General Dynamics. The APKWS concept was successfully demonstrated in September 2005 by British Aerospace (at which point financing was taken over by the United States Navy). Production of APKWS conversion kits started in 2010 and the first operational use of an APKWS Rocket occurred in Afghanistan during 2012.

1.1.4 The need to reduce cost

For a rocket to be effective in asymmetric warfare it must be accurate within a few meters over a range between 1 and 8 km – a requirement that a standard Free Flight Rocket launched from a helicopter cannot achieve for reasons as described later. The solution for this problem as promoted by the main weapon manufacturers of the world thus far was to control the path of the rocket after launch (a closed loop system using measurements from recent past as feedback).

One of the problems of asymmetric wars is that they tend to evolve into insurgencies that last for many years. This implies the use of large numbers of people, equipment and ammunition over an extended period of time. Such a scenario brings into play the enormous monetary cost of war. *Crawford* [16] analyzed the war efforts of the United States in Iraq, Syria, Afghanistan, Pakistan and abroad (Homeland Security) and concluded that it amounted to about $4,79 trillion for the period 2001 to 2017. Because wars are so expensive, they can be won on the battle ground but lost at the bank. "Smart Rockets" are indeed very accurate, but they each cost $28000 in 2012 terms according to *The Economist* [17]. This represents a stiff monetary penalty for asymmetric wars which are usually characterized by a multitude of small-value targets dispersed in time and space as prescribed by Mao Tse-Tung [3].

In performing Cost of Operational Effectiveness Analysis (COEA) some researchers point to the fact that an APKWS rocket represents only a third of the cost of a Hellfire missile. This is a small

comfort, due to the enormous cost of a Hellfire missile. The APKWS rocket simply does not fare well in providing a precision weapon that has a really low cost. According to *The Economist* [17] the price of an APKWS kit is about $28000 and the price of a Hydra-70 rocket fitted with an M151 High Explosive warhead about $1000 in 2012 terms. Hence, the APKWS kit implies a factor 28 increase in the price of a standard Hydra-70 rocket.

Note that the price increase of a Hydra-70 fitted with an APKWS kit versus a stardard Hydra-70 rocket is offset by the number of rockets required to eliminate a target. Such a comparison is performed by *The Economist* [17] who assumes a single target taken out by a single APHWS rocket would require two Rocket Launchers (each containing 19 standard MK-66 MOD4 rockets) to blanket the same target. Following a different approach to perform a Cost of Operational Effectiveness Analysis *Wonnacott* [9] assumes a standard MK-66 MOD4 rocket to have a one sigma dispersion (horizontal) of 12 milliradians/1000 meter downrange. His analysis indicates (amongst others) a cost of $294k to eliminate a large truck with 50% probability of kill using standard MK-66 MOD4 rockets versus $10k using the same rocket fitted with a Guidance Kit. Note however that *Wonnacott* [9] assumed the price of a standard MK-66 MOD4 rocket to be $1k and the same rocket fitted with a Guidance Kit to be $10k in 1997 terms. Reality shows that this ratio was overly optimistic with a factor 3 when compared with the same prices as quoted by *The Economist* [17] in 2012 terms.

1.1.5 The need to stay hidden

In their analysis of the Anaconda Mission (part of the Afghan war) and Medina Mission (part of the Iraqi war) *Cassidy* [6] and *Groenke* [7] illustrate the extreme vulnerability of the AH64D Apache Longbow Helicopter. What emanates from their studies is the fact that forces like the Taliban, el Quaeda and Iraqi Republican Guard studied Combat Helicopter doctrine and used this knowledge to succesfully engage the AH64D Apache Longbow Helicopter using simple World War II weapons such as AK-47 assault rifles, DUSHKA heavy machine guns and RPG-7 rockets.

Studying the use of the combat helicopter in the Afghan, Iraqi and Syrian wars reveals a common denominator – insurgent weapons such as those used during the Anaconda and Medina missions require the person to acquire the helicopter visually in order to engage it. This requirement is also applicable to shoulder-launched missiles such as the SA-7 and Stinger. It is becoming increasingly clear that the combat helicopter of the 2020's will have to make use of adverse weather (such as heavy mist and rain during night) to perform many of its missions. Provided a number of piloting prerequisites are met there can be safety in adverse weather because it can prevent insurgent forces from acquiring the helicopter visually, rendering World War II weapons as described above as well as shoulder-launched missiles such as SA-7 and Stinger useless.

As can be expected such an approach has some drawbacks, including the fact that the target cannot be designated because laser light cannot penetrate rain and mist. This renders optically guided weapons such as Hellfire and APKWS useless under these conditions. One solution for such a scenario would be the capability to launch standard Free Flight Rockets with very high accuracy in the first place.

1.2 An alternative approach **(Think of interesting questions)**

Can the objective of a precision weapon that is really low cost be satisfied in other ways? Two observations as described below suggest that there may indeed be an alternative approach.

1.2.1 Toss bombing on Cheetah C Jet

The Author was involved in a project during which the Inertial Navigation System (INS) and Fire Control Computer (FCC) of a Cheetah C jet fighter was modified by Denel Aeronautics during the period 2014 to 2015. The FCC contains a ballistic algorithm based on a Model Predictive Control Loop that manipulates the flight path of the aircraft as illustrated in Figure 7. The algorithm generates a set of flight directives that aspire to steer the aircraft into an optimal position at the release point, such that the survivability of the aircraft is maximized and the trajectory of the bomb passes through the target.

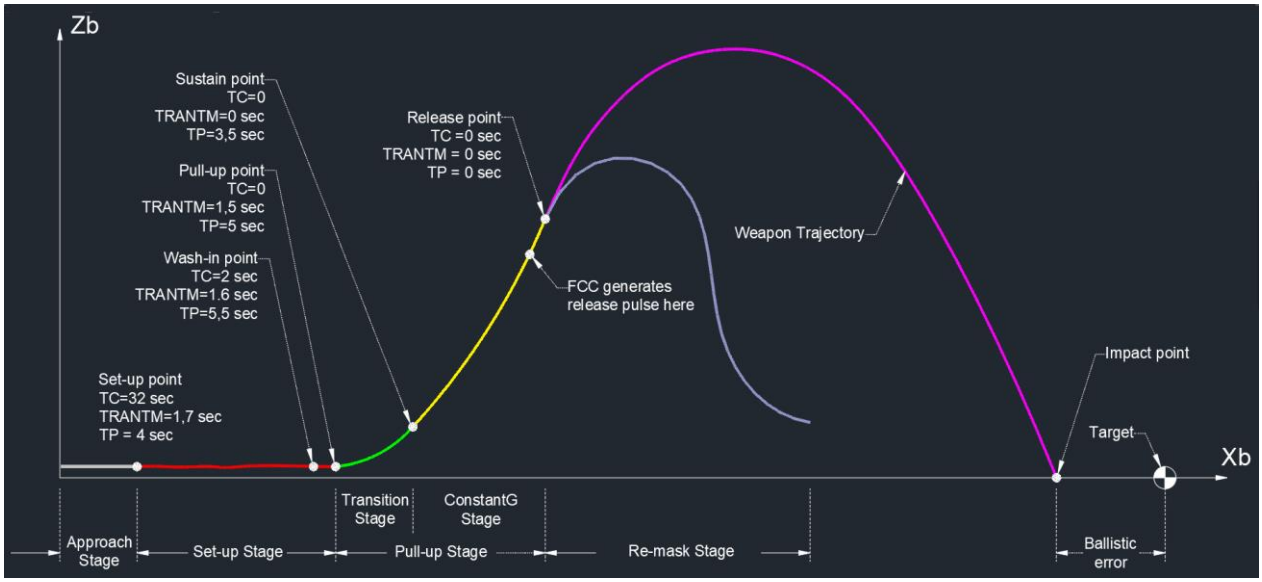


Figure 7. Pull-up flight profile during TOSS maneuver [A Landman, 2015]

Ensuring correctness of the ballistic model and using proper ballistic coefficient data yielded the toss bombing results as illustrated in Figure 8. Note that in this photo the circle is bare earth, 100 meter in diameter. During this test the aircraft approached from the left along the yellow line, releasing MK82 practice bombs from a distance of about 7 km. The spread in impact points orthogonal to the yellow line were dominated by the pilot and gave a cross range CEP of about 25 meter whereas the spread in impact points along the yellow line was determined by the ballistic model and gave a down range CEP of about 5 meter (hard to determine an exact value since the test comprised only 4 tosses).

Note: *The data represented in Figure 7 and Figure 8 is not available in the open literature. It was obtained by the Author during his involvement with the development of Cheetah C.*

Figure 8. Impact points of Cheetah Toss Test [Denel Aeronautics, 2016]

The implication of the Cheetah Toss Test is clear – understanding the dynamics of a ballistic problem, correcting for the error sources involved and consistently using properly measured ballistic coefficient data yields impressive results.

1.2.2 Accuracies achieved by Sniper Teams

Another scenario that sets a precedent for the PRLS concept is the accuracy achieved by snipers in the Afghanistan, Iraqi and Syrian wars. The latest of these kills was performed by a Canadian soldier using a McMillan Tac-50 rifle over a range of 3450 meter [18]. These accuracies are achieved by stringent control of things such as the mass, size and shape of the bullet as well as amount and composition of propellant in the chamber. Other considerations include Coriolis force, estimating local wind and maintaining the barrel in the correct orientation when the gun is fired as illustrated in Figure 9. Stringent control of all these things allows the sniper to predict the trajectory of the bullet with high accuracy and make suitable aiming adjustments.

Figure 9. US Marine Corps sniper team [Cpl. Ricky S Gomez, 2015]

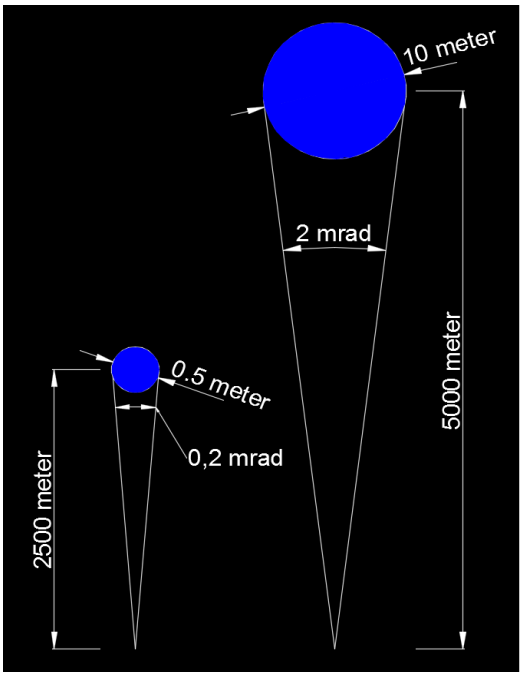


Figure 10. Accuracy comparison: Sniper Rifle versus PRLS [A Landman, 2017]

The accuracy of the PRLS as envisaged in this study and the accuracy as achieved by an average sniper using a sniper rifle are compared in Figure 10. It is clear that the accuracy achieved by the sniper is an order of magnitude better, so that it should be possible to achieve the PRLS accuracy as envisaged provided the aircraft does the same things as the sniper does. Like the sniper the aircraft can estimate the trajectory of the FZ90 Rocket beforehand and make suitable aiming adjustments taking into account things such as Coriolis force and local weather conditions that can be measured. Although production-related things such as the mass of the FZ90 Rocket, solid fuel composition and the quantity of solid fuel contained in the FZ90 Rocket Motor cannot be controlled, each FZ90 Rocket can be characterized as it moves down its Launch Tube. Suitable compensation for these errors can then be introduced before the FZ90 Rocket leaves its Launch Tube.

1.3 The PRLS Hypothesis **(Formulate Hypothesis)**

The main hypothesis of this study states that it should be possible to construct an airborne system containing a set of differential equations that describe the operation of a given helicopter-based Rocket Launching System (RLS) with good enough accuracy to predict the trajectory that a rocket launched from such a system would follow under a given set of flight conditions. It should be possible to perform this prediction with good enough accuracy to calculate an orientation for the rocket launcher that would result in the rocket trajectory reaching the target with very small error. It should be possible to achieve this by employing negative feedback in a Model Predictive Control Loop (MPCL) as illustrated in Figure 11. In this model, it should be possible to predict the response of the RLS one minute into the future (the maximum flight time of a FZ90 Rocket), thus creating a Model Predictive Control Loop with receding time horizon.

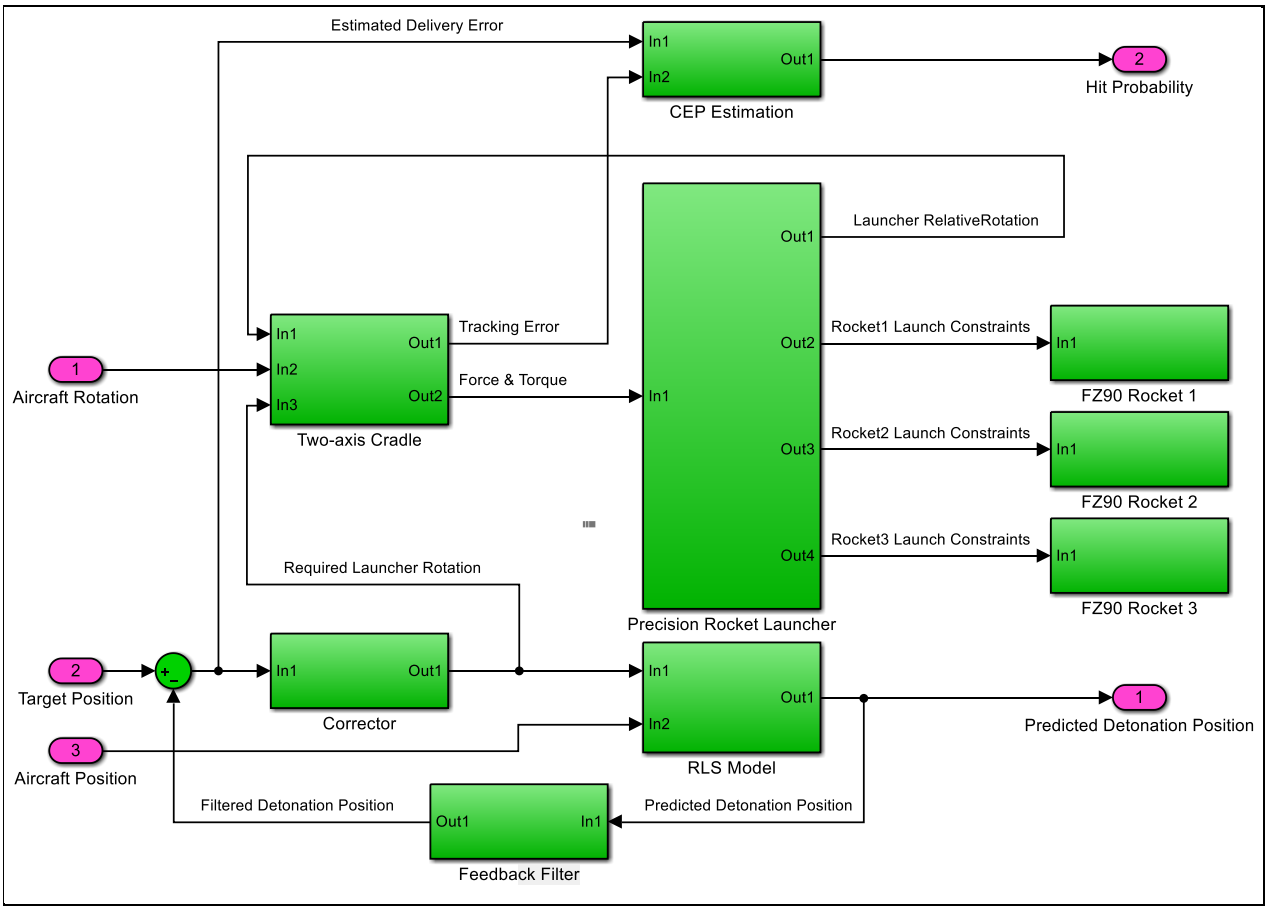


Figure 11. Model Predictive Control Loop for accurate rocket delivery [A Landman, 2014]

The Model Predictive Control Loop will be comprised of both software and hardware. The main software components will comprise the **Comparator, Corrector**, **RLS Model**, **Feedback Filter** and **CEP Estimation** blocks as illustrated in Figure 11. In the actual system these blocks will be software tasks running in real time within the Precision Rocket Launcher onboard the Aircraft. The main hardware components will comprise the **Two-Axis Cradle**, **Precision Rocket Launcher** and a number of **FZ90 Rockets** (only 3 shown here) as illustrated in Figure 11. The *Target Position* will originate from on-board devices such as the Main Sight or Inertial Navigation System (in the form of a navigational waypoint). The **RLS Model** block will calculate the *Predicted Detonation Point* of the selected rocket for the *Required Launcher Rotation* and *Aircraft Position* (as determined by Inertial Navigation System). This calculation will take into account various factors such as the characteristics of the FZ90 Rocket itself and downwash under the current flight conditions. The **Corrector** block will adjust the *Required Launcher Rotation* signal such that the *Error* signal tends towards zero. This can be made to happen at a suitably fast rate if the **Feedback Filter** and **Corrector** blocks are chosen correctly.

The the **Two-axis Cradle** will generate forces and torques that will rotate the **Precision Rocket Launcher** relative to itself, such that the difference between *Launcher Relative Rotation* signal (translated into Local Earth Coordinates) and *Required Launcher Rotation* signal tends to zero (i.e. a physical control loop). When this error becomes zero the theoretical and actual position and rotation of the **Precision Rocket Launcher** will be equal so that the **FZ90 Rocket** will hit the target if launched.

The *Error* signal in turn will be used by the **CEP Estimation** block to estimate the Hit Probability should the Rocket be launched under the current flight conditions. Hit Probability in turn, will be used as a gating signal to allow or inhibit launching of Rockets, communicated to the WSO or Pilot via suitable means (e.g. their Helmet Mounted Sight Displays).

1.4 Envisaged Performance (Develop testable predictions)

Contrary to the normal Scientific Method it is not possible to make formal predictions of PRLS performance at this stage of this study. To perform such predictions would require an overall **PRLS model** containing all the sub-models as developed in subsequent chapters of this study as well as sub-models of other contributing effects and mechanisms as identified in Chapter 8. Within this study, we limit the problem by only studying weather cocking due to main rotor downwash as the primary error mechanism. Having gained insight into this particular problem, we then perform a conceptual design of a system suitable for mitigating this particular problem, proving that at least one solution exists. Ultimately, an overall **PRLS model** would have to be built and a real PRLS system flown on a real aircraft to formulate proper predictions of PRLS performance. Nevertheless, we can arrive at subjective estimates of PRLS performance based on experience and performances of other systems as described before:

a) It should be possible to manipulate the launch conditions of the rocket such as to achieve an accuracy of 10 meter Circular Error Probable (CEP) on the ground (i.e. in a horizontal plane) at a range of 5 kilometer. It should be possible to achieve this throughout the flight envelope of the aircraft.

b) It should be possible to construct the PRLS within the mass, geometry and power constraints of the aircraft.

c) The computation throughput achievable with contemporary electronics should be adequate to implement the Model Predictive Control Loop of Figure 11 at a rate high enough to represent all the frequency components of the system in accordance with the sampling criteria of *Nyquist* [19].

d) It should be possible to replace the multiple recurring costs of launching Smart Rockets with a single non-recurring cost for modification of the aircraft to launch standard Free Flight Rockets instead. Break-even should be achieved within 50 Smart Rocket launches.

Note: *The purpose of this thesis is not to prove all aspects of the PRLS hypothesis. To do that will require multiple studies similar to this one. The ultimate objective of this study is to prove/disprove that the main rotor downwash of a helicopter can be calculated in real time using only equipment onboard the helicopter itself. This particular aspect forms part of items (b) and (c) above. Although various aspects of the PRLS hypothesis are addressed in this study, the PRLS Hypothesis in its entirety will be proven through a future study employing an overall PRLS Model in conjunction with flight test data as recorded on a real aircraft fitted with a real PRLS.*

Note: *The PRLS is a system of systems, and comprises both hardware and software components that have to be developed in a highly structured manner. It is also a safety-critical system that will be used to deliver rockets in close proximity to own forces, making a structured design and verification approach as defined in DO-178C [20] a necessity.*

1.5 Layout of the thesis

The layout of this thesis reflects a family of nested cycles of the Scientific Method. These cycles represent the research path followed by the Author in arriving at the PRLS concept.

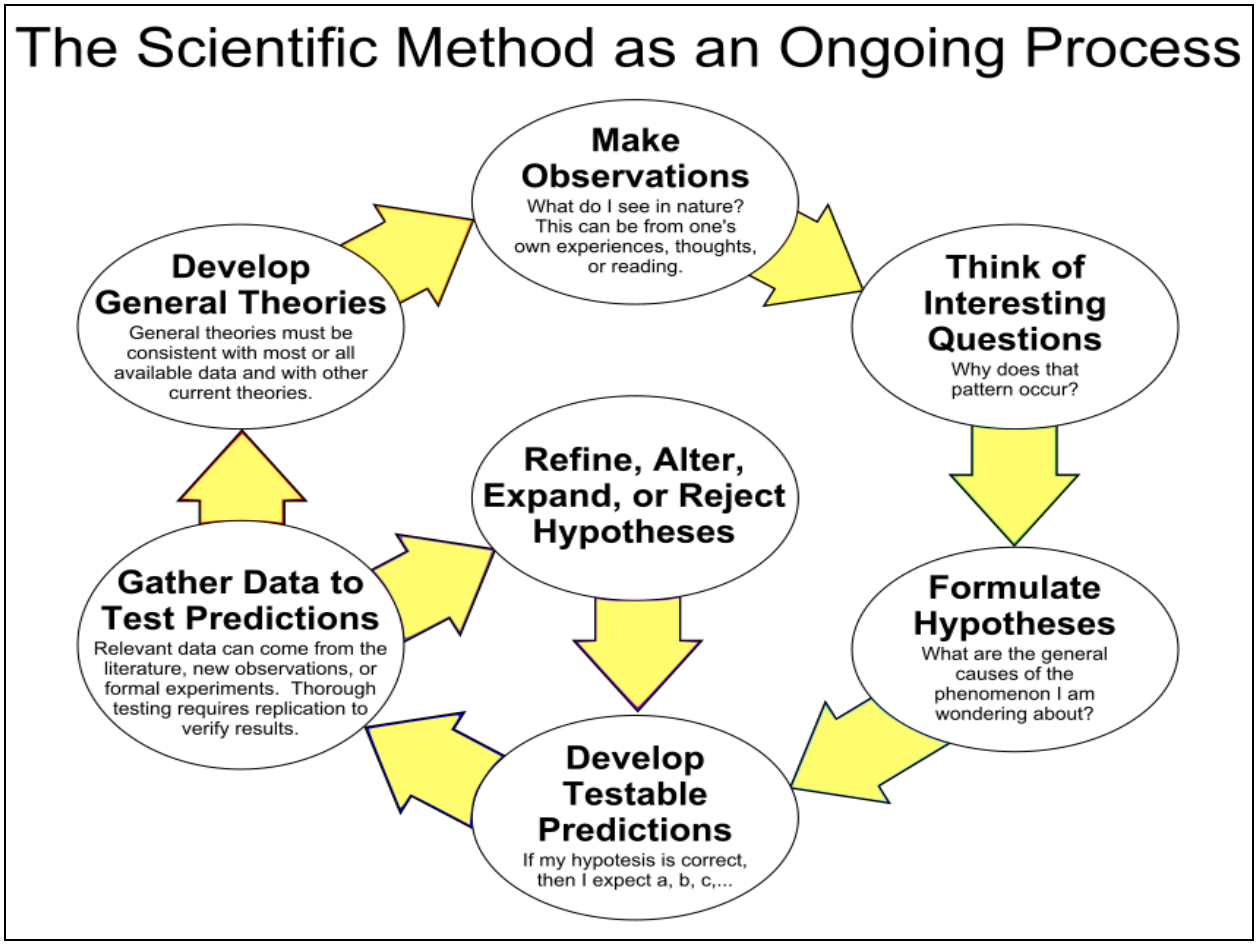


Figure 12. The Scientific Method Cycle [Wikipedia, 2017]

The steps of the Scientific Method as assumed in this study are those as identified by Wikipedia and illustrated in Figure 12. This same process is repeated for each chapter of this thesis. Note that these Scientific Method Cycles form a nested structure within which they influence each other. The process is therefore also iterative.

Chapter 1 works in conjunction with Chapter 8 of this study. Chapter 1 initiates the overall cycle of the Scientific Method which identifies the PRLS concept. It introduces the need for a low cost high precision helicopter based rocket launching system. Effectively this chapter is a representation of the operational research that would typically be performed by the Department of Defence of a country, resulting in a need to develop a cheaper solution because the cost of asymmetric war has proven overly expensive. Using observational evidence of existing systems Chapter 1 formulates the PRLS Hypothesis and associated predictions. The experiments required to test the PRLS Hypothesis are conducted in Chapters 2 through 7 and the General PRLS Theory is developed in Chapter 8 after considering the verification evidence produced throughout this study.

Chapter 2 uses the Scientific Method to investigate the so-called Coning Problem by enquiring which mechanisms may be at work to cause the coning and corkscrew effects as observed during qualification of the Rocket Launching System of Rooivalk AH-2A as well as RLS performance improvements observed under certain flight conditions. Here the primary observational evidence is in the form of long exposure photographs acquired during qualification of the Rocket Launching System of Rooivalk AH-2A.

Chapter 2 identifies interaction between rocket and downwash as the mechanism causing the observed behaviour. Coning is subsequently identified as one of the primary error terms of the RLS error budget on Rooivalk AH-2A. This conclusion and associated implications serve as one of the drivers for further cycles of the Scientific Method in subsequent chapters.

Chapter 2 also serves to develop a basic mechatronic model of the RLS system that is used throughout this study as well as within the PRLS itself. In performing this work, many of the building blocks to be used in the final PRLS are already determined at this stage of the study.

Chapter 3 uses the Scientific Method to develop a model of the Main Rotor that forms part of the mechatronic model of the RLS system. The Main Rotor Model is subsequently used to excite the Downwash Model from which the computational domain of the PRLS system is determined. This conclusion and associated implications serve as one of the drivers for further cycles of the Scientific Method in subsequent chapters.

Chapter 4 uses the Scientific Method to derive an algorithm (differential equation) suitable for estimating the main rotor downwash of a helicopter, such that it would be suitable for implementation on an Array Processor.

Chapter 5 uses the Scientific Method to postulate and test a design for the Precision Rocket Launcher. It identifies and evaluates the internal architectures of Central Processor Units (CPU), Graphic Processor Units (GPU) and Field Programmable Gate Arrays (FPGA) and the suitability of these devices in designing an Array Processor capable of surmounting the apparent limitation set by Amdahl's law. It also evaluates the solution in the light of other considerations such as commutation (variation of boundary conditions) and interrogation of the evolving solution in real time. Finally, it describes the use of the Simulink development environment and how it can be used to develop, test and program FPGA's as part of a larger system.

Chapter 6 uses the Scientific Method to postulate and test an Array Processor design called the Concurrent Model. It describes the test results achieved with this model and how the Concurrent Model was used to generate of a set of reference data for use during conversion of the Concurrent Model into fixed point notation. It also describes the conversion of the Concurrent Model from double precision to fixed point notation and finally the implementation of the Fixed Point Concurrent Model in FPGA, extracting relevant characteristics of the solution for comparison with other solutions.

Chapter 7 uses the Scientific Method to postulate and test an Array Processor design called the Pipeline Model. It describes the test results achieved with this model and how the Pipeline Model was used to generate of a set of reference data for use during conversion of the Pipeline Model into fixed point notation. It also describes the conversion of the Pipeline Model from double precision to fixed point notation and finally the implementation of the Fixed Point Pipeline Model in FPGA, extracting relevant characteristics of the solution for comparison with other solutions.

Chapter 8 consolidates the conclusions of all the chapters of the study and formulates a General PRLS Theory as required by the Scientific Method.

Appendices A, B and C contain a derivation of the ZY'X'' direction cosine matrix, the derivation of the **RigidElement2** block and a script written in Matlab for the Small Box Experiment of paragraph 4.5.5 and the Big Box Experiment of paragraph 4.5.8.

1.6 Summary (Develop General Theories)

This chapter was used to introduce the first half of a Scientific Method Cycle that forms part of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS). This was done by describing as main observation the advent of asymmetric warfare and concomitant need for a low cost high precision helicopter based rocket delivery system.

As additional observations the marriage between Rocket and Helicopter was shown to be an attempt of the aviation and weapons industry to solve the problem of dispersion of insurgent forces in time and space as teached by Mao Tse-Tung. By using test results of Rooivalk AH-2A it was shown that the performance of the standard version of this marriage is inadequate to avoid collateral damage. Also shown was that thus far the solution for the rocket accuracy problem promoted by the aviation and weapons industry has been the introduction of Smart Rockets (which use the Rocket and Target as parts of a control loop employing negative feedback using measurements from the recent past). Using additional observations and studies, the Smart Rocket was shown to be deficient in the light of:

- The need to reduce cost.
- The need to use adverse weather.

The question was then asked if contemporary electronics can be used to produce alternative solutions for mitigation of the problems as listed above. Using as observations the accuracy achieved by Cheetah C during a series of bomb toss exercises and the accuracy achieved by snipers using ordinary guns the PRLS hypothesis was formulated in which the errors associated with standard Free Flight Rockets are corrected beforehand by employing a Model Predictive Control Loop (i.e. using predicted measurements from the future).

Finally, the predictions as listed below were formulated for testing the PRLS hypothesis assuming such a system is installed on Rooivalk AH-2A:

- CEP of 10 meter at standoff range of 5 km, achieved throughout the operational envelope of the Aircraft.
- Conformance with the mass, geometry and power constraints of Rooivalk AH-2A..
- Break-even cost after 50 Smart Rockets.

The final step (Develop General Theories) of this overall Scientific Method Cycle is dependant on the test results of Chapters 2 through 7 and hence, is formulated in Chapter 8.

2 The Rocket Coning Problem (Cycle 2)

"A new type of thinking is essential if mankind is to survive and move toward higher levels". Albert Einstein

This chapter introduces the second Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. In order to implement the PRLS concept it is necessary that the error mechanisms resulting in the poor ballistic performance of standard Free Flight Rockets (FFR) as observed (refer paragraph 1.1.3) be understood. The purpose of this chapter is to investigate and model the first of the main error mechanisms referred to as the **Coning Problem**.

The effect of these mechanisms can be expressed in the form of an error budget that is useful for understanding the problem and directing the research and design effort.

The difference between trajectories of standard Free Flight Rockets launched from fixed wing and rotary wing aircraft are shown as primary observational evidence of the Coning Problem as illustrated in Figure 13. Additional evidence is presented in the form of a long exposure photographic image of Rooivalk AH-2A delivering a salvo of four FZ90 Rockets at night.

Figure 13. Skewing rocket trajectories during qualification of RLS on Rooivalk AH-2A [Wes Lindenberg, 2013]

Questions are asked as to what the mechanisms involved might be, resulting in the hypothesis that the main rotor downwash and some other related mechanisms interact in a complex manner to cause the coning effect as observed.

A multibody model referred to as the **Rocket Launching System Model** is developed to support the coning hypotheses. This model comprises a train of rigid bodies interconnected with various types of kinematic joints, one of which is Space (entity that surrounds all the Major Components of the PRLS such as the Fuselage, Main Rotor, M159 Rocket Launcher, FZ90 Rocket and

Target). Space is modelled as a relatively simple entity with few Degrees of Freedom (DOF) in this chapter. In subsequent chapters it is modelled with millions of DOF.

Coning of the FZ90 Rocket as revealed by stimulation of the **Rocket Launching System Model** with a simple downwash model is proposed as testable predictions of the Coning Problem. The characteristics of the phenomena as predicted with the **Rocket Launching System Model** are compared with evidence gathered from free literature as well as photographic evidence gathered during the Rooivalk AH-2A Qualification campaign.

The main contribution to the knowledge database made in this chapter is quantification of the magnitude of the Coning Problem and its relative size as part of the error budget. No model or data that can be used to quantify the Coning Problem directly could be found in the open literature. In fact, the open literature appears to have either underestimated the effects of downwash or overlooked it altogether.

2.1 Straight & Skewed Trajectories (Make observations)

The trajectories of Free Flight Rockets launched from both a fixed-wing aircraft and hovering helicopter are illustrated in Figure 14. It is evident that the trajectories of rockets launched from a fixed-wing aircraft flying at high speed are straight and well-behaved while the trajectories of rockets launched from a rotary-wing aircraft in hover are skewed with a pattern that appears to be random.

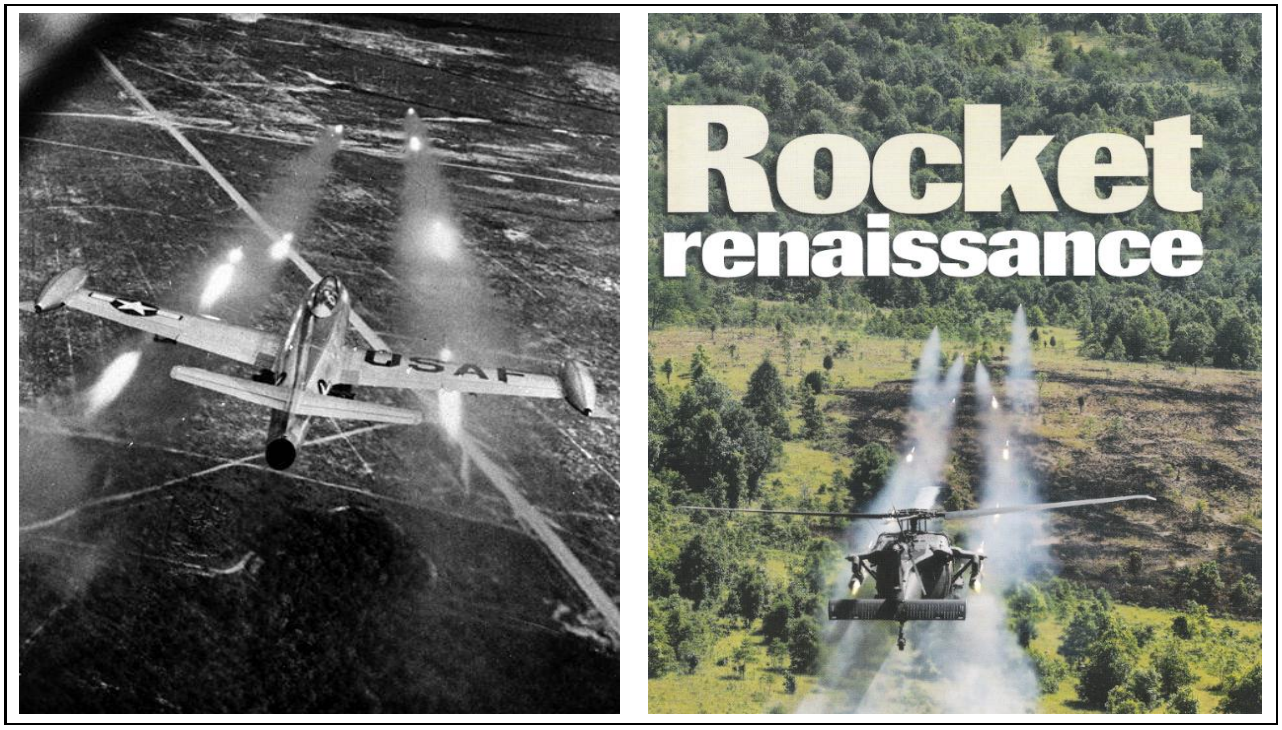


Figure 14. Rockets launched from a F-84E jet and a Black Hawk helicopter [Wikipedia: 2014 and Defense Helicopter, 2014]

The effect of this random movement on the footprint of rocket impacts with the ground as viewed through the Main Sight during qualification of the Rocket Launching System of Rooivalk AH-2A is illustrated in Figure 5. During these tests the aircraft was hovering 200 feet above ground at a range of 2,5 kilometers. Hence, the dimensions of the ground footprint formed by these rocket impacts was about 300 meter wide by 3000 meter long.

Figure 13 provides photographic evidence of the perturbation of Free Flight Rockets launched from Rooivalk AH-2A in the hover. It is evident that the rockets were launched in pairs (as in Figure 16) and that each pair saw similar type of perturbations.

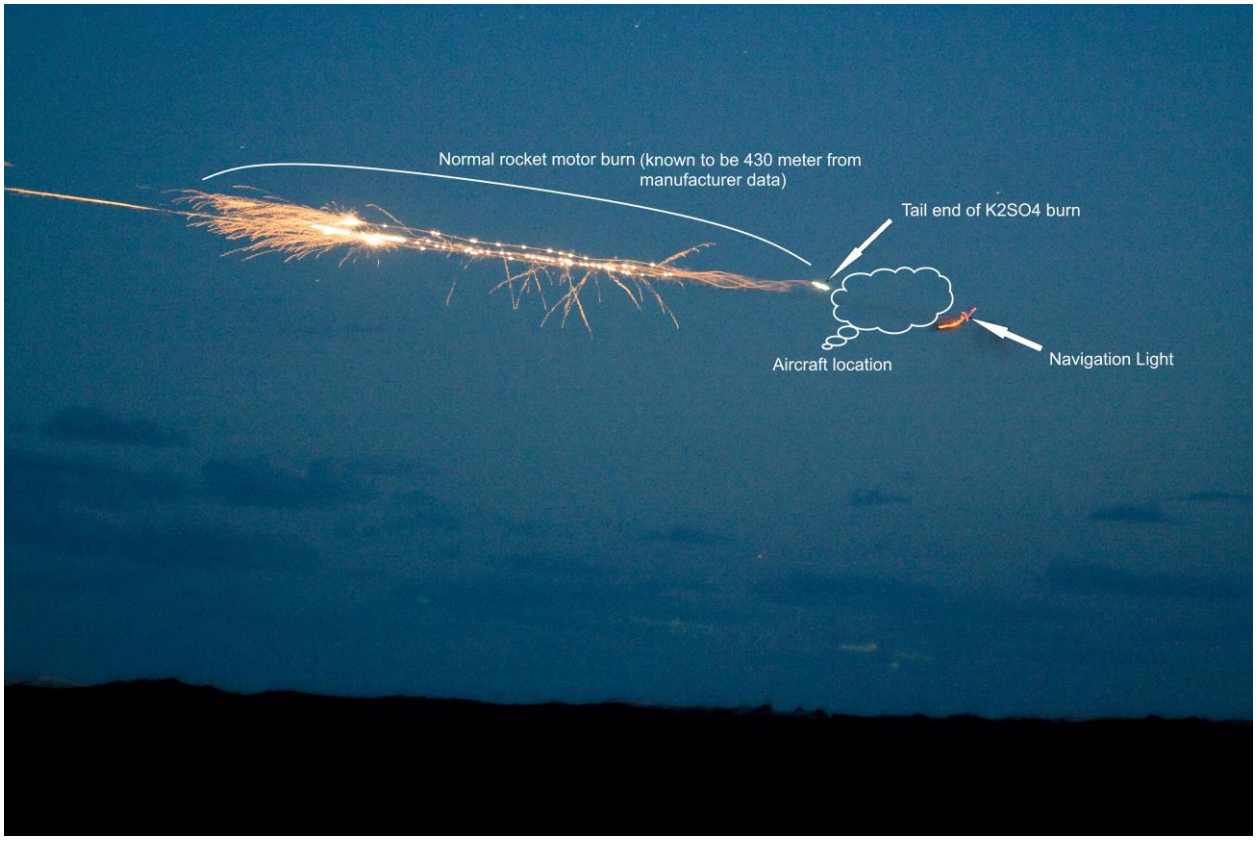


Figure 15. FZ90 Rockets coning on Rooivalk AH-2A [Wes Lindenberg, 2013]

Figure 16. Range of the K_2SO_4 Stage [Wes Lindenberg, 2013]

Figure 15 is a long exposure photograph of Rooivalk AH-2A delivering a salvo of four FZ90 Rockets at night. In this image the camera was located to the left of the Aircraft which is invisible because it was moving slowly during the delivery. The paths travelled by the nozzles of the rockets are indicated by the orange lines which show that the rockets performed a coning or corkscrew motion that decreased with distance travelled. This type of motion is reported by *Berner*, *Abate* and *Dupuis* [22] where the disturbance was caused by a side moment. The red object is the Navigation Light on the tail of the Aircraft (which moved while the camera shutter was open). The white "object" is part of the K_2SO_4 Stage of the third or fourth rocket that was still in progress when the camera shutter opened (the rockets were launched in pairs as illustrated in Figure 16). The K_2SO_4 burn marks the very early stage of the launch process during which additional oxygen is released to avoid secondary combustion in the rocket exhaust which can cause an engine flameout as described by *Hawley* [11]. The process ends when the FZ90 Rocket is about 10 meter in front of the Aircraft as illustrated at bottom of Figure 16. The extending orange lines represent the normal operation of the FZ90 Rocket Motor where the K_2SO_4 rod has been consumed.

Pilots operating Rooivalk AH-2A in the Democratic Republic of the Congo (DRC) report that the width of the footprint of a salvo of rocket impacts with the ground narrows with speed (reaching about 50 m CEP at a range of 3 km with the aircraft travelling at 80 knots True Air Speed).

Note: *Figure 13, Figure 15 and the observation of reduction of footprint width versus aircraft speed as observed while launching FZ90 Rockets from Rooivalk AH-2A are not available in the open literature. This information was obtained by the Author during his involvement with the development and operation of Rooivalk AH-2A.*

Dahlke and *Batiuk* [21] present aerodynamic data including spin-up sequence of the MK-66 MOD1 Rocket Motor in conjunction with the M151 high explosive warhead and M261 submunition warhead. This data can only be used to characterize a MK-66 Rocket model since it was determined in the absence of downwash.

Mermagen and *Oskay* [23] measured the spin history of a MK-66 MOD1 Rocket Motor combined with modified warhead containing a yawsonde test installation. The rockets were launched from a ground-borne launcher. A tendency to yaw was detected at spin reversal and minimum spin. Late flight instability was detected on two rockets out of four. This data can only be used to characterize a MK-66 Rocket model since it was determined in the absence of downwash.

Daniels and *Hardy* [24] investigated Catastrophic Yaw - a type of motion associated with slender bodies fitted with fins. Although this type of motion may indeed occur in case of the FZ90 Rocket it is not the motion under consideration here.

2.2 Why the Coning? **(Think of interesting questions)**

Why would the trajectories of rockets launched from a fixed-wing aircraft flying at high speed be straight and well behaved while the trajectories of rockets launched from a rotary-wing aircraft in the hover be skewed? The open literature does not provide a clear answer to this question other than piecemeal information that has to be assembled into a concise hypothesis.

Evident immediately is that the main mechanism at work in Figure 14 cannot be play between rocket and rocket launcher, since the rockets were launched from rocket launchers in both cases. Inspection of Figure 14 shows that in the case of the F-84E Jet the rockets were launched into air that has not been disturbed by the aircraft while in the case of the Black Hawk Helicopter they were launched into air that has been set in motion by the main rotor. It would appear that the initial launch conditions of rockets launched from the helicopter were disturbed, and that these disturbances were amplified due to the butterfly effect as described by *Lorenz*

[25]. Note that the butterfly effect is applicable here since the power transferred to the air by the hovering Black Hawk helicopter would have been about 3 megawatt. For such a large power transfer the downwash must have transitioned into chaotic behavior as shown by *Taylor* [26] as well as *Swinney* and *Gollub* [27].

Wonnacott [9] quantifies this random movement of rockets as a one sigma dispersion (horizontal) of 12 milliradians/1000 meter downrange in the case of the MK-66 Rocket fitted with an M151 Warhead. He does not indicate if the dispersion results with the rocket exposed to downwash during launch or not. In addition, he does not model the interaction between components such as rocket, rocket launcher, main rotor and downwash nor does he mention any dependency of dispersion on helicopter speed of travel.

Lee, *Choi* and *Kwon* [28] calculated the perturbation of the trajectory of a Free Flight Rocket during the early stages of launching such a rocket from a hovering UH-60 helicopter by using a Finite Element Model combined with a 6 DOF rocket model. Their analysis shows substantial perturbation of the rocket trajectory during the first 70 meter of travel, but makes no conclusion as to the overall effect of these perturbations on the rest of the trajectory. It should also be noted that the rocket used in the model of *Lee*, *Choi* and *Kwon* does not spin since it uses a set of steering canards. They solve the Euler equations solely to calculate the steering performance of such a rocket.

Note: *Since the model of Lee, Choi and Kwon [28] calculates the trajectory of the rocket over a distance of 70 meter we conclude that the dimensions of their computational domain or "Flight Box" was probably 140m x140m x140m cube (assuming aircraft to be at centre). We will return to this topic in paragraph 4.5.1.*

MIL-HDBK-762(MI) [29] describes a wide range of error mechanisms including coning and wobble (dynamic imbalance) as well as interactions between components such as rocket, rocket launcher, main rotor and downwash. However, it makes no specific calculations to quantify these effects for the FZ90 Rocket or the MK-66 Rocket launched from a helicopter. Note however that *MIL-HDBK-762(MI)* does calculate an error budget for a rocket launched from a ground based rocket launcher in the absence of downwash. Some of these terms are indeed applicable to the case considered here.

Breaux [30] shows that the ballistic algorithm of the COBRA combat helicopter considers the effect of downwash, even though the algorithm does not model the gyroscopic effects of the spinning rocket.

2.3 The Coning Hypothesis **(Formulate Hypothesis)**

Having considered the available evidence, we conclude that a helicopter remains in the air by using a main rotor to pump a large quantity of air towards the ground. The result is a huge jet of fast-moving air known as the main rotor downwash. Due to the body plan of the modern combat helicopter, the main rotor downwash also occupies that area in space where rockets launched from the helicopter start their operational lives. The interaction between the main rotor downwash and a rocket being launched is major because the velocity of the air in the main rotor downwash is comparable to the velocity of the rocket when it leaves the launch tube. Upon clearing the launch tube the rocket therefore tends to fly upwards while underneath the main rotor because the Centre of Gravity of the rocket is located behind its Centre of Mass (the FZ90 Rocket and MK-66 Rocket both have a high stability margin). This weather cocking of the rocket represents a change in direction which causes the spinning rocket to precess (cone) in order to conserve its angular momentum. As the rocket travels it is accelerated by the rocket motor further still, causing it to travel faster and spin more rapidly (i.e. gain more angular momentum) which causes the amplitude of the precession to taper off. This "coning" of the rocket causes a

large angle of attack between rocket body and air which leads to a big increase in drag. This changes the start conditions of the rocket which are subsequently amplified over the rest of the trajectory due to the butterfly effect.

The description of the coning problem above assumes a hovering helicopter flying under Out of Ground Effect (OGE) conditions. In such a case the main rotor downwash takes the shape of a doughnut centred around the main rotor (we will return to this topic in paragraph 4.5.9). With the helicopter in forward flight the donut skews towards the rear of the helicopter so that at some speed the downwash skews over the Stub Wing, causing the rockets to be launched into clean air with concomitant reduction in dispersion as observed. In the case of Rooivalk AH-2A this condition is reached at a speed of 60 knots, so the width of the ground footprint as reported by pilots flying in the DRC must be caused by the remainder of error mechanisms other than coning. It is clear that rocket coning must be a major error mechanism, since removal thereoff reduces the width of the ground footprint from 300 meter to 50 meter.

2.4 Coning Predictions (Develop Testable Predictions)

The first prediction that can be tested is the spin rate of the FZ90 Rocket upon exit from the M159 Rocket Launcher. This should be reasonably close to the spin rate of the MK-66 MOD1 Rocket Motor in conjunction with the M151 High Explosive Warhead under similar conditions. According to *Dahlke* and *Batiuk* [21] as well as *Mermagen* and *Oskay* [23] the spin rate and speed of the MK-66 Rocket Motor fitted with an M151 High Explosive Warhead should be about 10 Hz upon exit from a M159 Rocket Launcher.

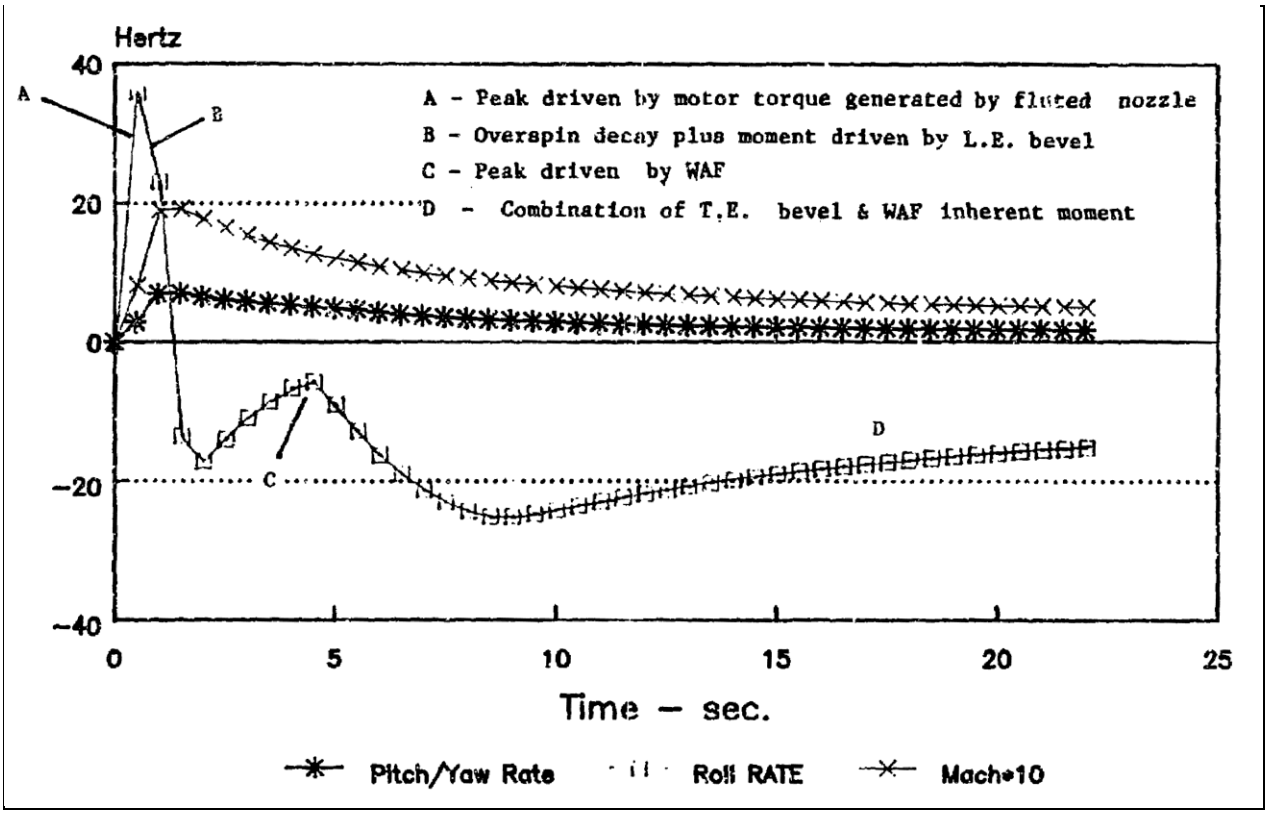


Figure 17. Typical evolution of roll, pitch, yaw and Mach number for MK-66 Rocket [Dahlke and Batiuk, 1990]

Note: *Refer to Figure 56 for a description of Wrap Around Fin, Leading Edge Bevel and Trailing Edge Tab.*

The second prediction is the maximum spin rate and maximum speed of the FZ90 Rocket. Again these parameters should be reasonably close to the same of the MK-66 Rocket motor fitted with M151 High Explosive Warhead. According to *Dahlke* and *Batiuk* [21] the typical spin rate and speed of a MK-66 Rocket Motor fitted with an M151 High Explosive Warhead as measured with yawsonde Flight Test Installation (FTI) is as illustrated in Figure 17. This sets the maximum speed of the FZ90 Rocket at Mach 1,9 (i.e. 646 m/s at +15 °C) and its maximum spin rate at 32 Hz.

The third prediction is the maximum coning offset and coning angle reached by the FZ90 Rocket during the launch stage. In Figure 15 the distance between the launcher and end of launch stage (plume of sparks on left in Figure 15) is about 430 meter[1]. Using this distance as reference and using a ruler to measure the ratio of coning diameter to launch distance, the coning diameter at end of K_2SO_4 stage was estimated to be about 10,75 meter. The coning angle was estimated to be 21° as illustrated in Figure 18.

Figure 18. Estimating Coning Angle [A Landman, 2017]

The above predictions are tested in paragraph 2.5.9 using simulation with the **Rocket Launching System Model.**

[1] Authors data as obtained from Forges de Zeebrugge.

2.5 The RLS Model (Gather Data to Test Predictions)

Given the Coning Hypothesis of paragraph 2.2 the Rocket Launching System of Rooivalk AH-2A can be modelled using the **Rocket Launching System Model** is illustrated in Figure 20. The objective of this model is to predict the interaction between the Major Components of the Rocket Launching System (RLS) of Rooivalk AH-2A including FZ90 Rocket, M159 Launcher, Cradle, Actuator, Stub Wing, Fuselage, Main Rotor and Space. The arrangement of these components on Rooivalk AH-2A is illustrated in Figure 4 and Figure 19.

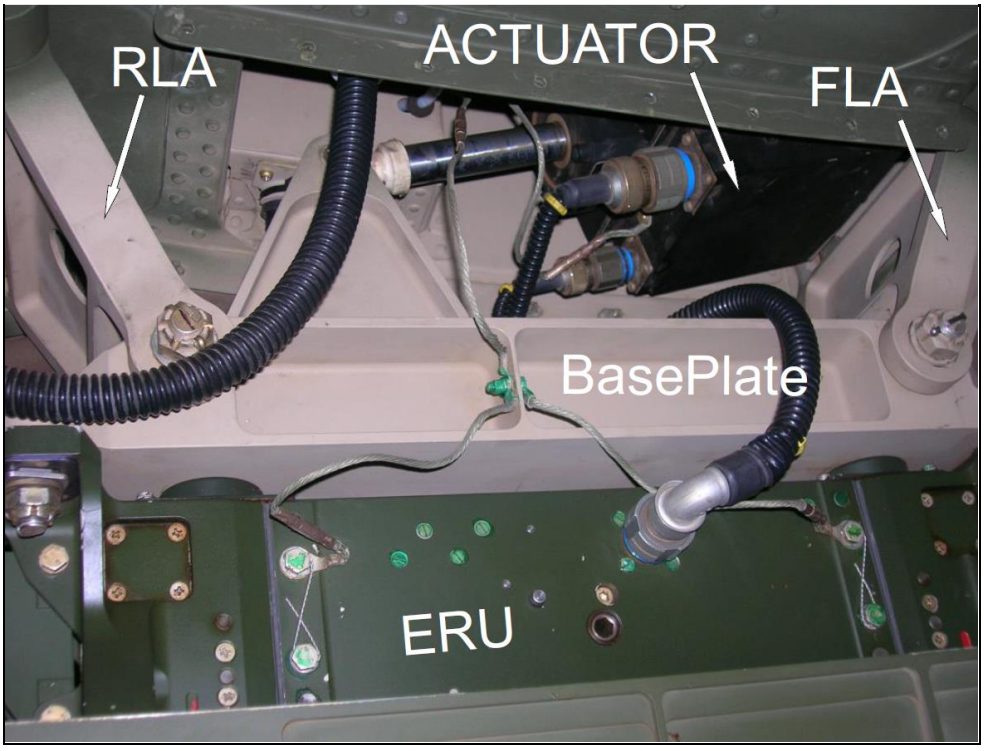


Figure 19. Cradle, Actuator and ERU fitted on Rooivalk AH-2A [A Landman, 2014]

Partial builds of the **Rocket Launching System Model** can be constructed to evaluate only certain aspects of the Rocket Launching System of Rooivalk AH-2A. This allows the model to be expanded in stages (i.e. as required by the various Scientific Method Cycles of this study). In this chapter a partial build comprising only the **FZ90 Rocket** block, **Rocket-Launcher I/O** block and **Space** block is built. This is adequate for evaluating the interaction between FZ90 Rocket, M159 Rocket Launcher and Downwash during the launch stage as well as the subsequent coast stage until reaching the Target.

Note that Space is represented by the air in immediate vicinity of the aircraft (i.e. Near Flow Field or Main Rotor Downwash) as well as the air relatively far away from the Aircraft (i.e. Far Flow Field or Atmosphere).

2.5.1 Use of Engineering Multi-Ports

As shown in Figure 20 the **Rocket Launching System Model** is a multibody model comprising a train of rigid bodies interconnected with various types of kinematic joints. The data flows follow the convention of engineering multi-ports as described by *Karnopp*, *Margolis* and *Rosenberg* [31]. Following this convention, all rigid bodies respond to force inputs by generating shape outputs while all kinematic joints do the opposite. Bond Graphs are not used in this study because they are not supported by Simulink.

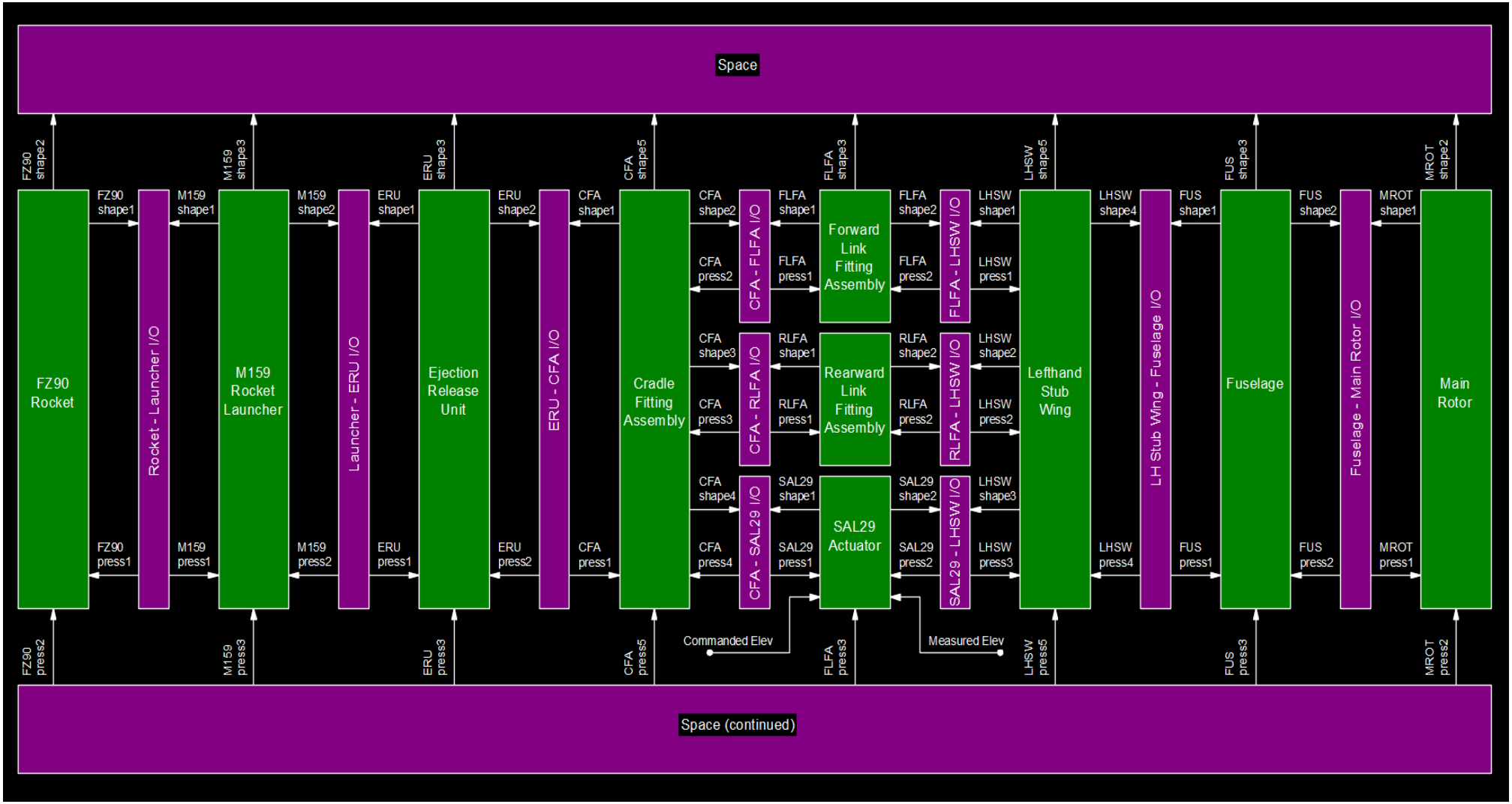


Figure 20. Data Flow Block Diagram of the Rocket Launching System Model [A Landman, 2017]

2.5.2 Real Time Operation

Apart from modelling the Rocket Launching System of Rooivalk AH-2A for performance evaluation purposes, the **Rocket Launching System Model** of Figure 20 is also intended to evolve into the **RLS Model** block of Figure 11. The **Rocket Launching System Model** must therefore be designed to run in real time. Ultimately, it will be compiled into C or HDL for installation on DSP, GPU or FPGA devices within the Precision Rocket Launcher.

The only example of such a real time application that could be found in the open literature is the ballistic algorithm of the BELL TEXTRON COBRA helicopter as described by *Breaux* [30]. This model views the rocket as a point mass (i.e. 3 Degrees of Freedom) and represents the downwash as a step function with constant velocity, anti-parallel to the gravity vector and within a radial range of 6,279 meter (i.e. below the main rotor disk). Within the Breaux Model the magnitude of the downwash is chosen in the range 3,2 to 13,716 m/s depending on flight conditions.

Note: *The downwash velocities used by the ballistics model of Breaux correspond with those of a 5 ton helicopter. In the case of an 8 ton helicopter like Apache AH64D Longbow or Rooivalk AH-2A the velocity of the downwash is in the region of 30 m/s (we will return to this topic in paragraphs 2.5.8.4, 3.5.5, 4.1 and 4.5.9).*

2.5.3 Use of lumped-elements

Another feature shared by the rocket models of *Wonnacot* [9], *Brown* [32] and *MIL-HDBK-762(MI)* [29] is that they are all lumped-element models of electromechanical systems as described by various authors including *Woodson* and *Melcher* [29] as well as *Karnopp*, *Margolis* and *Rosenberg* [31]. Representing a complex system with lumped-elements reduces and fixes the number of state variables required by the model and hence, the computational power required to run the model, especially if it has to run in real time.

This is the same simplification as described by *Kokalari et al* [28] for modelling the Cardiovascular System. *Kokalari* stresses that the complexity of a model should be chosen such as to match the accuracy required. *Woodson* and *Melcher* [29] describe modelling as a "*largely intuitive process*". *Karnopp, Margolis and Rosenberg* [31] describe modelling as "*The art and science of system modelling has to do with the construction of a model complex enough to represent the relevant aspects of the real system but not so complex as to be unwieldy*".

Finding a suitable balance between accuracy and speed has therefore been a quest that influenced every model developed under this study. In the case of the **Rocket Launching System Model** the model solves an inherently complex differential equation which cannot be calculated by hand. Judgements on the accuracy of the model must therefore be made on the basis of comparing simulation and empirical results.

2.5.4 Use of standard building blocks

One of the principles followed in developing the **Rocket Launching System Model** was the use of standard building blocks within the Simulink environment. This is the preferred development route followed in this thesis because it makes for easy construction of extensive models that would otherwise require a complex process of mathematical analysis and software coding.

Two standard building blocks were developed for the purposes of this study – the **Rigid Element** block and the **Gimbal Element** block as illustrated in Figure 21. Both blocks were written in C and packaged as an S-function so they could be precompiled for use in the Simulink environment. This avoids the time penalty of their piecemeal execution by the Simulink interpretor at run-time.

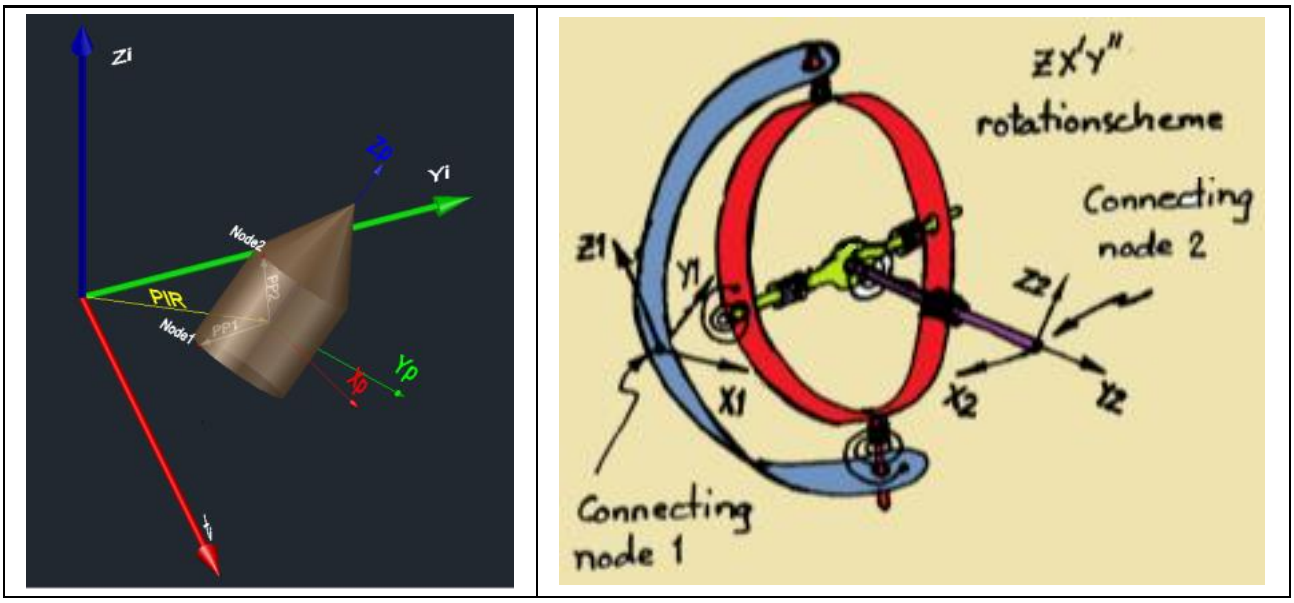


Figure 21. The Rigid Element block and Gimbal Element block [A Landman, 2017]

Use of the **Rigid Element** block and the **Gimbal Element** block allow for the construction of any multibody system comprising any combination of rigid bodies and kinematic joints including revolute, translational and cylindrical joints as described by *Flores* [36]. The use of Simulink also makes available a range of continuous and discrete solvers that can be chosen and configured to match the problem at hand avoiding the need to construct a dedicated solver as described by *Flores* [36].

Note: *Simulink contains a toolbox called Simscape which contains blocks similar to the **Rigid Element** block and the **Gimbal Element** block. Although the Simscape blocks can certainly be used, we wrote our own blocks because it allows us to modify and optimize the source code. For example, it allowed us to generate a number of derivatives of the **Rigid Element** block with various numbers of connecting nodes and physical characteristics such as mass and Moments of Inertia that can be controlled through external signals (e.g. the **Variable Mass Body** block of the **FZ90 Rocket** model).*

Standard building blocks developed under this study use engineering multi-ports as described by *Karnopp, Margolis* and *Rosenberg* [31]. In the case of rigid bodies the *effort* input is represented by *force* signals and the *flow* output by *position* signals. In the case of kinematic joints the *effort* input is represented by *position* signals and the *flow* output by *force* signals. Imposing this constraint was necessary because the mechatronic systems studied under this study are causal – any single block cannot determine both cause and effect. Adopting the convention as defined above makes for the easy construction of complex systems of rigid bodies and kinematic joints to represent objects such as a main rotor blade or a stub wing.

2.5.5 Coordinate systems

Unless otherwise stated, all equations, nodes and internal states of the **Rocket Launching System Model** are described in Celestial or Inertial Axes as described by *Wertz* and *Larson* [35]. Inertial Axes is a right-handed coordinate system with the **Z**i axis aligned with the spin axis of the Earth pointing towards the Celestial Pole. The **X**i axis lies in the equatorial plane of the Earth and points towards Vernal Equinox. The **Y**i axis is formed by the righthand rule. Although this coordinate system is not strictly stationary with respect to the stars, *Wertz* and *Larson* [35]

describes it as "sufficiently inertial" or subject to accelerations so low in comparison with those of the ***Rocket Launching System Model*** *that it can be assumed as inertial for the purposes of this study.*

Note: *The **Rocket Launching System Model** is described in Inertial Axes so that the laws of Newton and Euler can be used without modification. This assumption is not valid in Local Earth Axes which is a rotating geocentric system that induces pseudoforces such as coreolis and centrifugal acceleration. Pseudoforces are taken into account in the **Rocket Launching System Model** by transforming vectors (such as wind speed) from Local Earth Axes into Inertial Axes.*

In some cases use is made of Local Coordinate systems that are aligned with some physical features of the Rocket Launching System. In these cases transformation of vectors between different axes systems is achieved through multiplication with suitable Direction Cosine Matrixes, usually assuming a ZX'Y'' transformation scheme.

2.5.6 The FZ90 Rocket block

This block forms part of the **Rocket Launching System Model** as illustrated in Figure 20. The objective of the **FZ90 Rocket** block was to provide a real time model of the FZ90 Rocket comprising FZ90 Rocket Motor, FZ71 High Explosive Warhead and M423 Impact Fuze.

None of the Free Flight Rocket models available from the open literature except that as described by *Breaux* [30] are realtime applications, so a new model tailor-made for use in the **Rocket Launching System Model** was developed under this study. The algorithms as used by the models available through the open literature provide little clues as to how a real time rocket model should be constructed. These algorithms were nevertheless studied and their associated simplifying assumptions investigated in order to gain a better understanding of the problem at hand.

Note: *The ballistic algorithm of the Cheetah C Jet is another example of a Model Predictive Control loop running in real time. It cannot be used here because the information is not available through the open literature.*

Wonnacott [9] describes a 6 DOF rocket model to perform a trade-off study between standard Hydra-70 MK-66 rockets fitted with and without a guidance kit. This model does not simulate the effect of main rotor downwash or the interactions between FZ90 Rocket and M159 Launcher because it is intended to estimate the interaction between rocket, guidance kit and target. *Wonnacott* [9] avoids estimation of the interactions between FZ90 Rocket and M159 Launcher by initializing his model with predefined flight conditions, most of which correspond with ideal flight conditions a few seconds into the launch sequence.

Another problem with the Wonnacot Model is that, although it was written in Matlab, it was not intended for real time operation. Should it be used for modelling the FZ90 Rocket it would have to be converted into C and packaged as an S-function for use in the Simulink environment.

Brown et al [32] describe a 6 DOF rocket model written in FORTRAN and intended for predicting the trajectory of a non-spherical body around a non-spherical planet, within or above the atmosphere. This is a comprehensive model intended for space-borne applications that favours accuracy and not realtime operation. In addition, the model does not include the features necessary to model the peculiarities of the FZ90 Rocket, or the effect of main rotor downwash or the interactions between FZ90 Rocket and M159 Launcher.

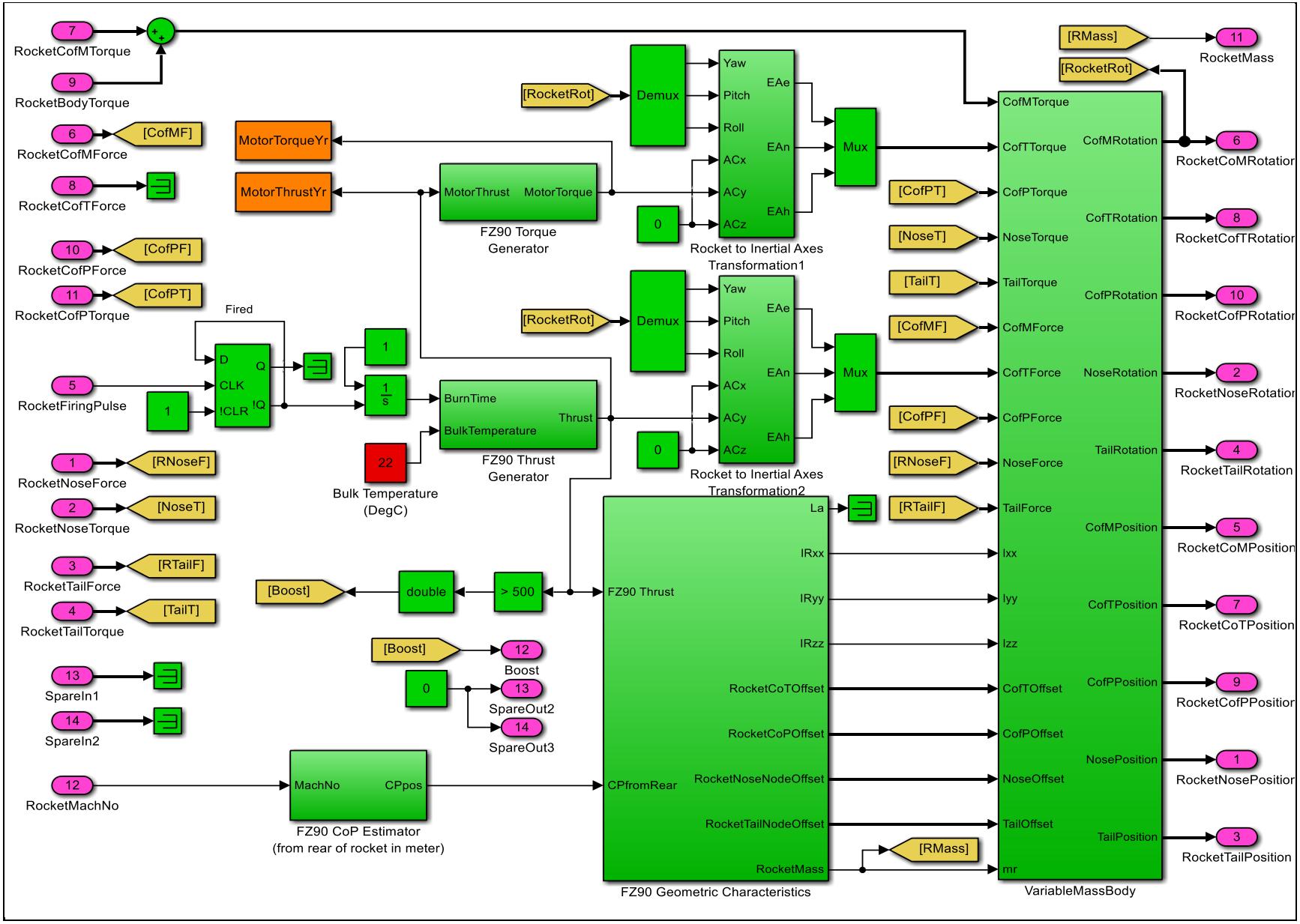


Figure 22. DFBD of the FZ90 Rocket block [A Landman, 2014]

MIL-HDBK-762(MI) [29] contains a 6 DOF rocket model together with a wide range of factors to be considered in the design of a solid fuel rockets. It describes a wide range of simplifying assumptions that can be used to simplify the equations of motion of a Free Flight Rocket depending on the particular application. It also contains a qualitative description of the interactions between rocket and rocket launcher, but does not provide a specific model or set of equations to simulate these interactions.

What the models of *Wonnacot* [9], *Brown* [32] and *MIL-HDBK-762(MI)* [29] have in common is that they all consider both the translational and rotational behaviour of the rocket under study. This supports our assumption that the **Rocket Launching System Model** must be able to model both the translational and rotational behaviour of the FZ90 Rocket to explain the coning effects as observed while launching the FZ90 Rocket from Rooivalk AH-2A.

The Data Flow Block Diagram of the **FZ90 Rocket** block is illustrated in Figure 22. It comprises five discrete blocks as shown, invented and configured in such a way as to yield a combined model suitable for modelling the FZ90 Rocket as used on Rooivalk AH-2A and running at iteration frequency of 10 kHz.

2.5.6.1 The Variable Mass Body block

This block is an adaptation of the **Rigid Element** block standard building block as developed by the Author. It was written in C (to maximize speed of execution) and packaged as an S-function for use in the **Rocket Launching System Model** (which is written in Simulink). Refer to Appendix B for the theory of the **Variable Mass Body** block.

The **Variable Mass Body** models the translational dynamics of the rocket as a rigid body in accordance with Newton and the rotational dynamics of the rocket in accordance with Euler as described by *Spiegel* [38]. The block accepts five forces and five torques acting through the Centre of Mass, Centre of Torque, Centre of Pressure, Nose and Tail of the rocket. It also accepts three (variable) Moments of Inertia around the three principal axes of the rocket as well as a (variable) mass located at Centre of Mass. The block generates as output the position and rotation of the Centre of Mass, Centre of Torque, Centre of Pressure, Nose and Tail of the rocket. All positions, rotations, forces and torques fed to the block or generated by it are expressed in Inertial Axes assuming a ZXY rotation scheme. Singularities in the Euler equations are checked and corrected for by the software. This implementation of the **Variable Mass Body** block allows calculation of the rocket trajectory while considering effects such as variation in mass, shift in Centre of Mass, shift in Centre of Pressure and changes of moments of inertia as the solid fuel of the rocket is consumed during launch. This arrangement automatically solves the *Tsiolkovsky* equation (movement of body that generates its own trust by expulsion of matter) as described by *Wertz* and *Larson* [35].

2.5.6.2 The FZ90 Thrust Generator block

The **FZ90 Thrust Generator** block estimates the thrust generated by the FZ90 Motor. The thrust force acts through the Centre of Thrust which is located somewhere within the nozzle of the FZ90 Rocket as function of mass flow and ablation of the nozzle throat. This location should really be determined through empirical test but no such information could be found in the open literature except an approximate indication of the position by *Hawley* [12]. Until proper information becomes available the position of the Centre of Thrust was therefore taken as located on the rocket centre line, 50 mm into the rear end of the nozzle.

Described in various handbooks such as *Wertz* and *Larson* [35] as well as *Campbell* and *McCandless* [37] the thrust F_t generated by the FZ90 Rocket Motor is governed by the rocket thrust equation:

$$F_t = \dot{m}_e V_e + A_e(P_e - P_\infty)$$ **Equation 1**

with $\dot{m}_e$ the mass flow rate through the nozzle, V_e the gas exit velocity with reference to the rocket itself, A_e the surface area of nozzle exit, P_e the pressure at nozzle exit and P_∞ the ambient pressure in vicinity of the nozzle. For the FZ90 Rocket Motor the left-hand term of the thrust equation dominates with the righthand term making a smaller contribution as function of nozzle shape.

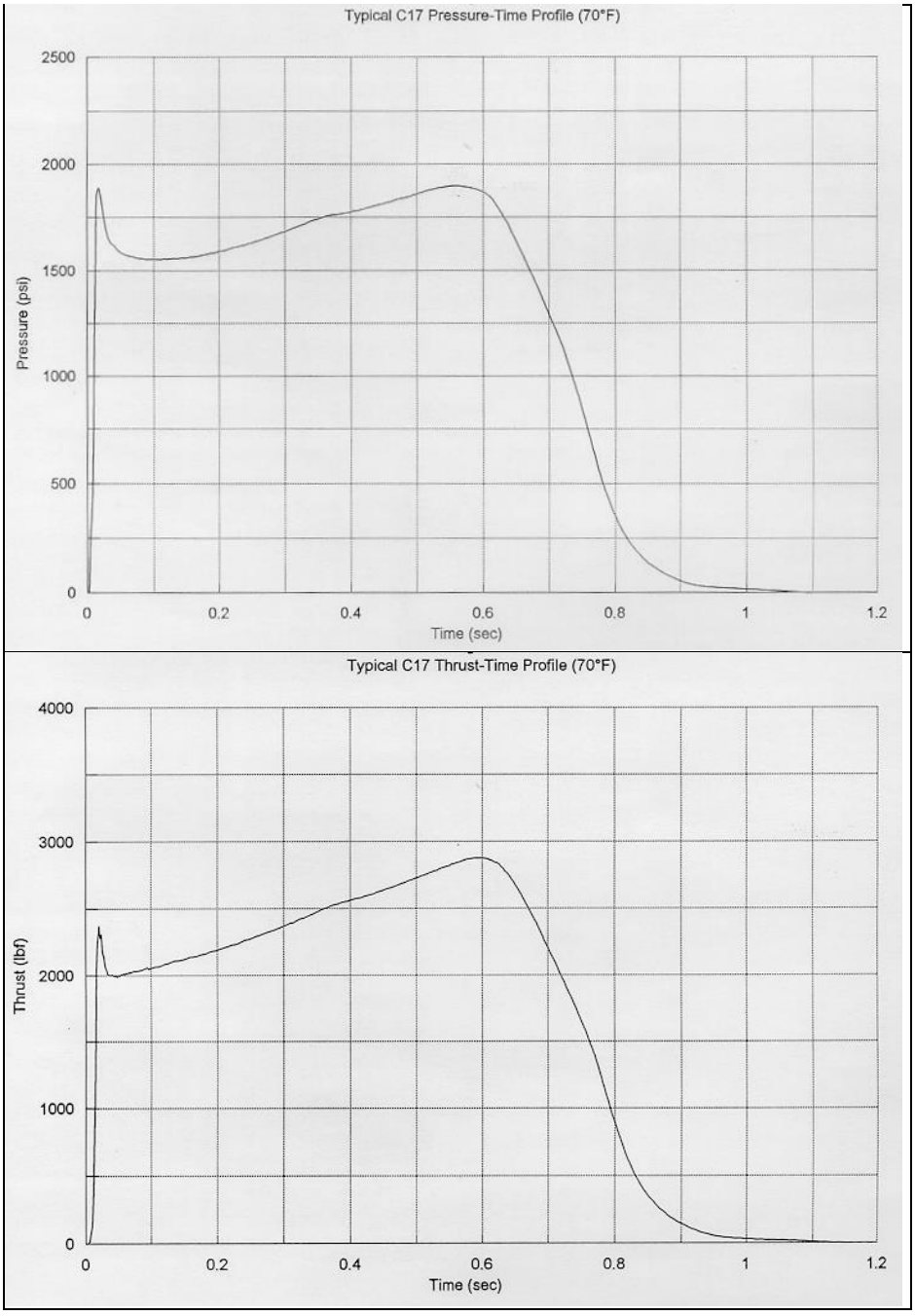


Figure 23. Evolution of pressure and thrust of C17 Rocket Motor [Bristol, 1998]

After ignition the pressure and temperature within the combustion chamber increases rapidly as illustrated in Figure 23. This causes the gas flow through the nozzle to increase until the speed of the gas flow in the throat equals the speed of sound. Under these conditions the throat becomes choked so that the increase in gas flow ceases as described by *Coulson* and *Richardson* [39]. Under these conditions the choked mass flow rate $\dot{m}_{ec}$ stabilizes at:

$$\dot{m}_{ec} = \frac{A_t P_t}{\sqrt{T_t}} \sqrt{\frac{\gamma}{R}} \left(\frac{\gamma + 1}{2} \right)^{-0.5\left(\frac{\gamma+1}{\gamma-1}\right)}$$ **Equation 2**

with A_t the throat area, P_t the chamber pressure, T_t the chamber temperature and γ the ratio of specific heats of the gas inside the combustion chamber. Because A_e, T_t and γ are essentially constant for the FZ90 Rocket Motor, the evolution in thrust is directly proportional to the combustion chamber pressure which increases as the burn face increases. Since the height change of the FZ90 Rocket trajectory is relatively small, pressure P_∞ is essentially ambient pressure. Exit pressure P_e depends on the internal shape of the nozzle which can result in overexpansion, underexpansion or perfect expansion as described by *Campbell* and *McCandless* [37].

It is clear that although the thrust generated by the FZ90 Rocket Motor can be determined with Equation 1 and Equation 2, some terms are unknown. For example, the evolution in combustion chamber pressure depends on the initial shape (round or star-shaped) of the solid fuel drill-hole, burn rate catalysts in the fuel mixture and bulk temperature of the solid fuel as described by *MIL-HDBK-762(MI)* [29]. Another problem is ablation of the throat area (made of carbon) during launch. It is clear that the thrust curve for the FZ90 Rocket Motor should rather be based on empirical results. *Šaulys* [41] describes a test setup to measure the thrust generated by some unknown solid fuel rocket motor. The shape of the thrust curve that he measures is in agreement with the basic shapes of the FZ90 Rocket Motor (as measured by Forges de Zeebrugge), the C17 Rocket Motor (as measured by Bristol) and the MK-66 Rocket Motor as described by *Dahlke* and *Batiuk* [21].

Using the thrust types as defined by *Hill* and *Peterson* [42] we note from Figure 24 that the thrust curve of the C17 Rocket Motor is strongly progressive (probably due to a bore-hole with circular geometry) while the thrust curve of the FZ90 Rocket motor is only slightly progressive (probably due to a drill-hole with star-shaped geometry). In contrast, the thrust curve of other rocket motors like the AeroTech NAR [40] is regressive (a desirable quality during stage separation when low acceleration levels are required to ensure the two halves seperates gradually without any collisions). It would appear that the FZ90, C17 and MK-66 rocket motors were chosen progressive by design. A gradual increase in thrust during launch might be necessary to avoid rattle or friction between rocket and launch tube, but the precise reason behind this choice is currently not clear.

Note: *The thrust and pressure curves of the FZ90 and C17 Rocket Motors as illustrated in Figure 23, Figure 24 and Figure 25 are not available in the open literature. They were generated from data as measured by Bristol and Forges de Zeebrugge and acquired by the Author during the course of his work at Denel Aeronautics*.

It is evident that there is substantial variation between the thrust curves of similar rockets made by different manufacturers. The correct curve to use for the **FZ90 Thrust Generator** block is thus clearly the thrust curve as measured for the FZ90 Rocket Motor, but a problem remains in that the variation in thrust of the FZ90 Rocket Motor due to variation in bulk temperature is unknown. The thrust curves of the C17 Rocket Motor as measured at bulk temperatures of +71°C, +22°C and -54°C were however available to the Author. Integration of the C17 thrust curves shows that the total impulse of the C17 Rocket Motor remains constant within 2,8% for bulk temperatures in the range -54°C to +71°C. The small variation in total impulse is probably due to reduction in efficiency of chemical reaction at lower bulk temperature. By scaling the thrust curves at +71°C, +22°C and -54°C along the time axis to yield the same burn time and fitting a cubic equation to the data the Author found that the variation can be predicted by scaling the horizontal axis (Time) of the C17 thrust curve with constant A_s calculated as follows:

$$A_s = 1{,}3.10^{-7} T_b^3 + 2{,}1.10^{-5} T_b^2 + 0{,}0018\, T_b + 0{,}95$$ **Equation 3**

with T_b the bulk temperature of the solid fuel in degrees Celcius. In similar fashion the vertical axis (Force) of the C17 thrust curve must be scaled with constant B_s calculated as follows:

$$B_s = 1{,}4.10^{-7} T_b^3 + 2{,}1.10^{-5} T_b^2 + 0{,}0020\, T_b + 0{,}95$$ **Equation 4**

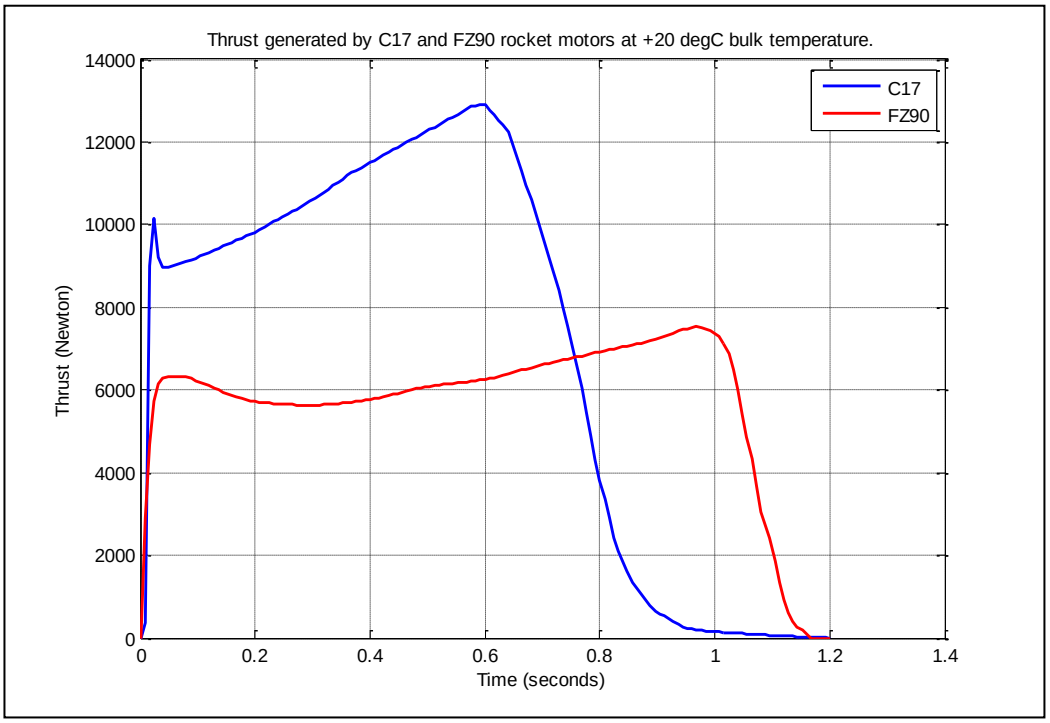


Figure 24. Thrust curves of C17 and FZ90 rocket motors [Bristol and Forges de Zeebrugge, 1998]

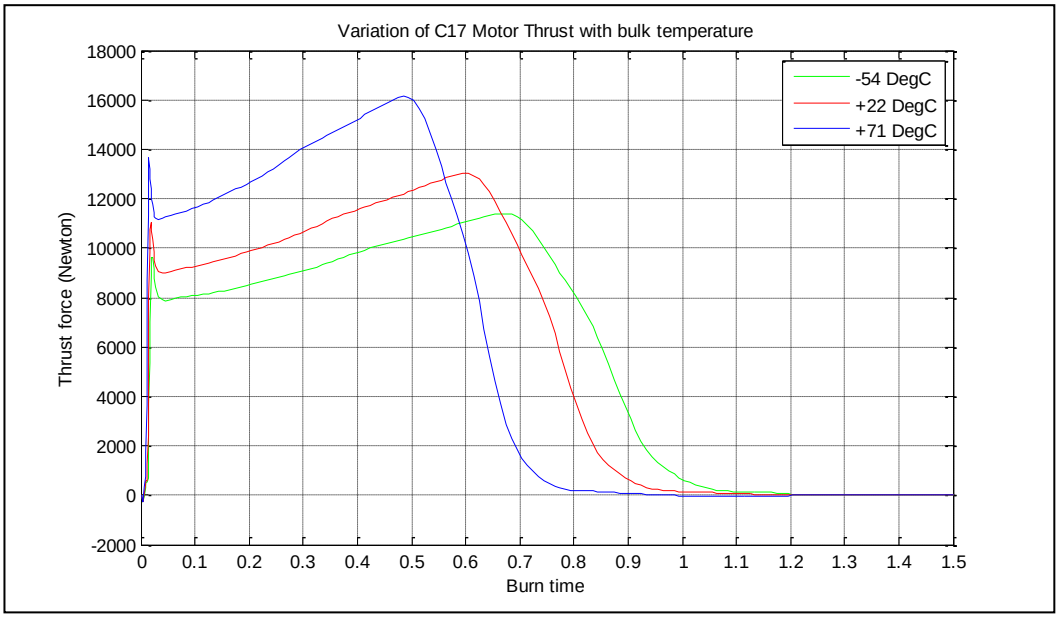


Figure 25. Thrust curves of C17 Rocket Motor versus bulk temperature [Bristol, 1998]

Since the FZ90 and C17 rocket motors are of similar design, the assumption is made that the thrust curve of the FZ90 Rocket Motor can be scaled with the same factors to account for variation in bulk temperature of the solid fuel.

2.5.6.3 The FZ90 Torque Generator block

The **FZ90 Torque Generator** block estimates the torque generated by the FZ90 Motor. This torque acts through the tail node of the FZ90 Rocket Motor. The torque is generated by slots in the case of FZ90 and MK-66 rocket motors and vanes in the case of C17 Rocket Motor, all located in the rear section of the Nozzle as illustrated in Figure 26.

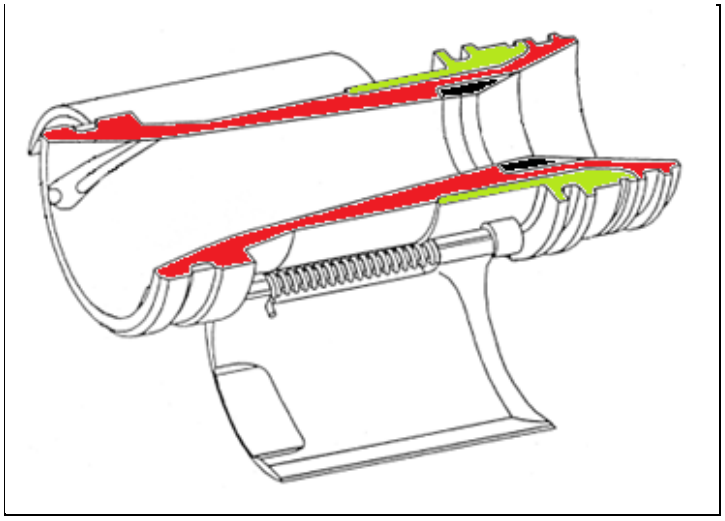


Figure 26. Cutaway view of C17 Nozzle showing single vane and wing [Bristol, 1998]

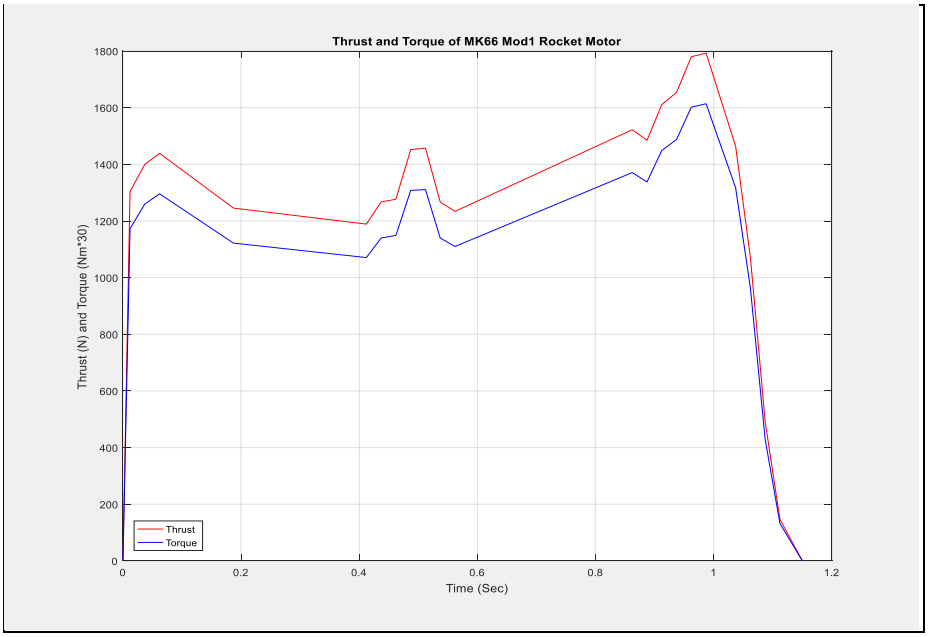


Figure 27. Thrust and Torque curves of MK-66 Rocket Motor [Dahlke and Batiuk, 1990]

Since the detailed internal shape of the nozzle of FZ90 Rocket Motor was unknown to the Author, the motivation for using a torque curve determined through empirical means also applied here. However, a torque curve for the FZ90 Rocket Motor was not available, so it had to be

derived using the torque curve of the C17 Rocket Motor as reference shape (acquired by the Author during the course of his work at Denel Aeronautics).

As an alternative the torque curve of the MK-66 Rocket Motor as described by *Dahlke* and *Batiuk* [21] was also considered as reference shape. This presented a problem because the thrust and torque curves of the MK-66 Rocket Motor as described by *Dahlke* and *Batiuk* have exactly the same shape as illustrated in Figure 27. It is evident *Dahlke* and *Batiuk* assumed the torque of the MK-66 Rocket Motor to be directly proportional to its thrust. This assumption is incorrect as demonstrated by the differences between thrust and torque curves of the C17 Rocket Motor as illustrated in Figure 25 and Figure 29. An actual measurement of the torque produced by the MK-66 MOD2 Rocket Motor as described by *Kim* and *Walbridge* [10] is illustrated in Figure 28. It is evident that the torque of the MK-66 MOD 2 Rocket Motor rises exponentially (within about 100 msec) to a torque of 3,2 ft-lbf that essentially remains constant for the duration of the burn.

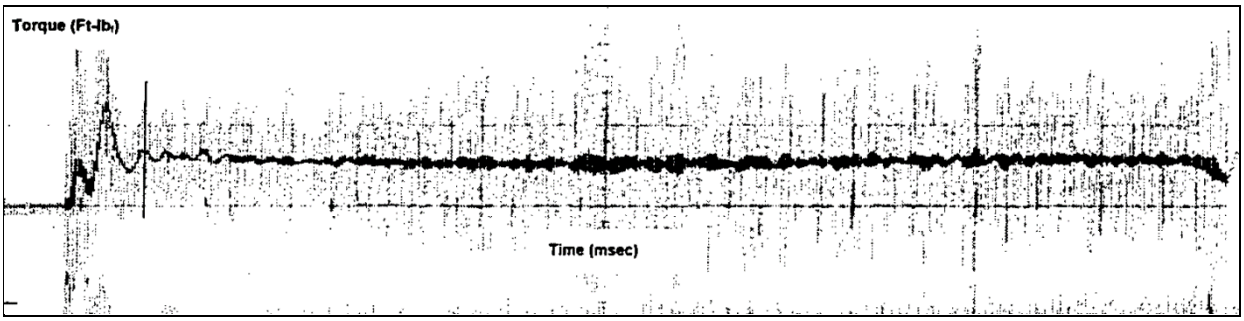


Figure 28. Torque curve of MK-66 MOD2 Rocket Motor at 77 °C [Kim and Walbridge]

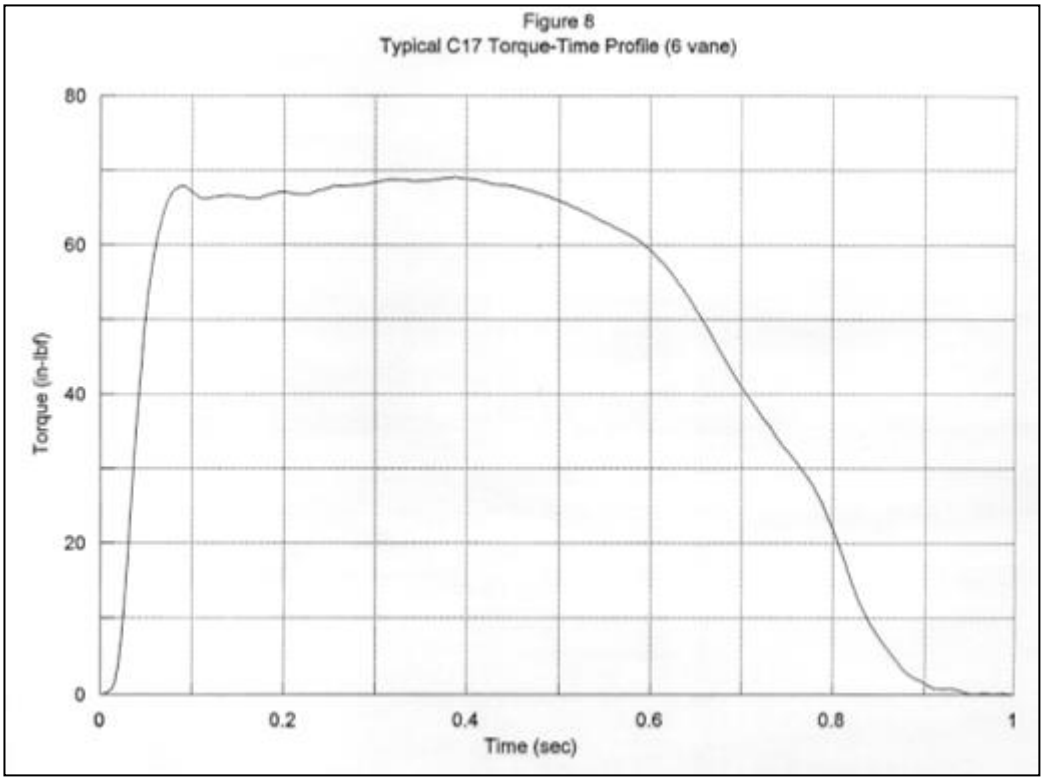


Figure 29. Torque curve of C17 Rocket Motor at 20 °C bulk temperature [Bristol, 1989]

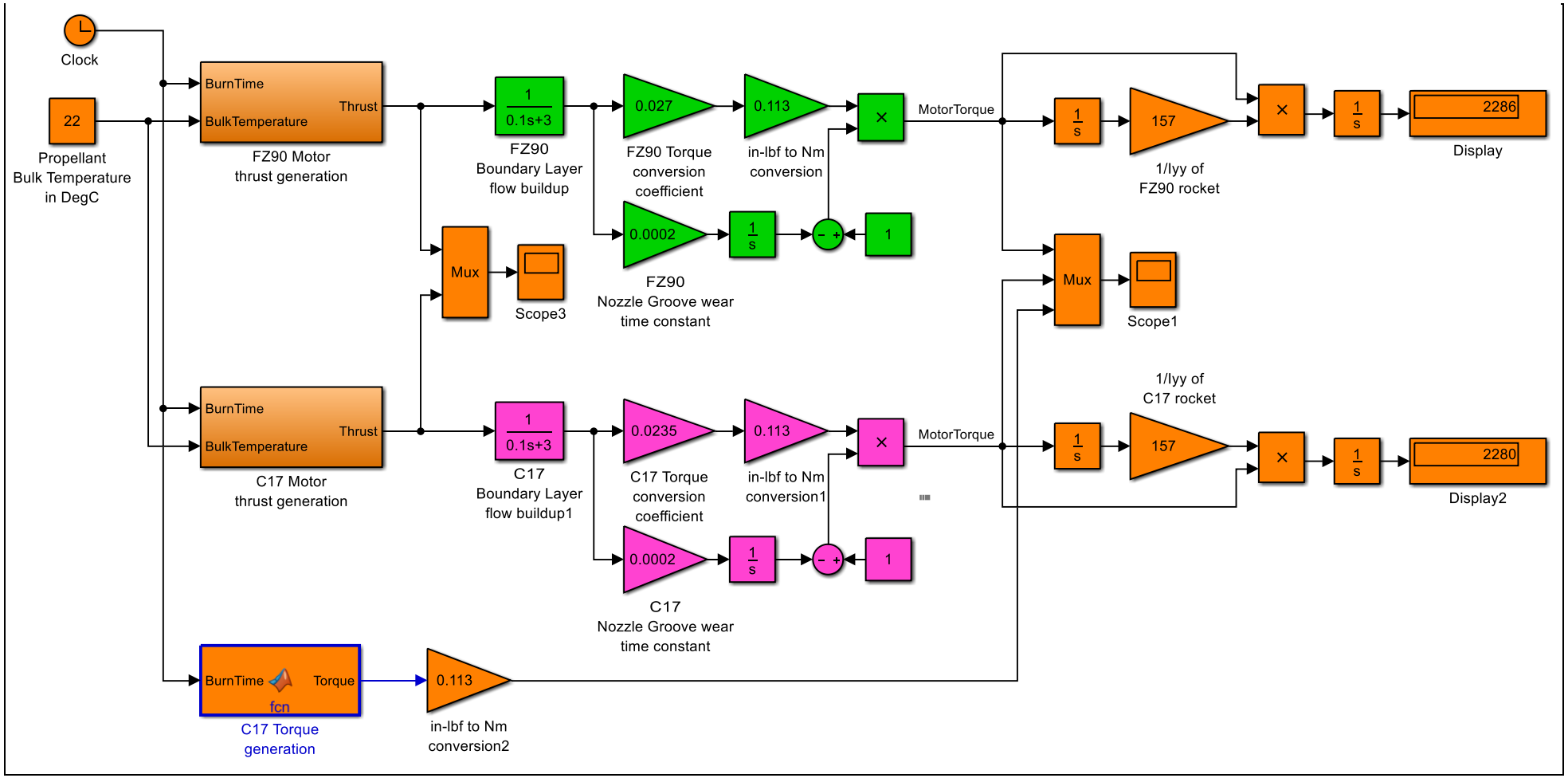


Figure 30. Estimating Ablation Time Constant and Torque Conversion Coefficient for FZ90 Rocket Motor [A Landman, 2014]

It is evident that the torque generated by both the C17 and MK-66 MOD2 rocket motors lag the thrust generated by the motors with about 0,1 seconds. Why would this be? **Hypothesis:** The gas in the throat of the C17 nozzle becomes choked within 30 ms as illustrated in Figure 23. As the choked gas in the throat moves down the nozzle, it has a Mach Angle smaller than the slope of the Nozzle, creating a subsonic layer which speeds up relatively slowly in an exponential fashion (momentum transferred via sliding layers of gas above), thus initiating the torque much later.

Additional support for the subsonic layer hypothesis is to be found in the fact that the torque curve of the C17 Rocket Motor remains relatively flat while its thrust curve increases rapidly as the burn face of the combustion chamber increases. The reason for this "buffering" is that the thickness of the boundary layer remains constant while the gas in the boundary layer remains subsonic as described above. Hence, the mass flow through the flutes remains constant even though the mass flow through the throat of the nozzle increases.

Another effect evident from the thrust and torque curves of the C17 Rocket Motor is the fact that the torque curve starts to diminish after 0,4 sec burn time while the thrust curve climbs for 0,6 sec. No information on the mechanism that causes this behaviour could be found in the open literature. It might be the result of a shift in shock wave pattern within or at the nozzle exit that might be induced by the increase in mass flow rate as the solid fuel is consumed.

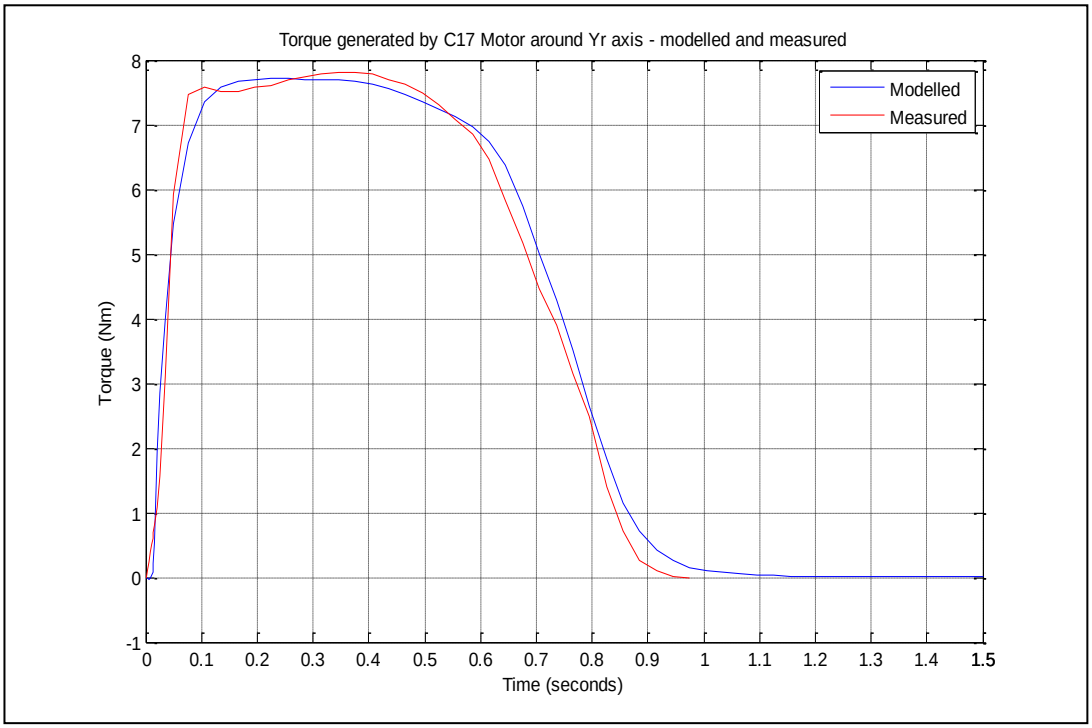


Figure 31. Actual and modelled torque curves for C17 Rocket Motor [A Landman, 2014]

This postulated process as described above can be modelled as illustrated with the green blocks in the DFBD of Figure 30. The block on left simulates the exponential build-up in speed gas in the subsonic layer. The result is then scaled – similar to the method of *Dahlke* and *Batiuk* [21] – to yield the basic torque of the C17 Rocket Motor (i.e. throat undamaged). Depreciation in torque is modelled by integrating a fraction of the speed of the boundary layer gas and scaling the basic torque with the remainder. The Depreciation Time Constants as shown in the DFBD of Figure 30 were determined through trial and error, yielding the actual and modelled torque curves for the C17 Rocket Motor as shown in Figure 31.

To determine the FZ90 Torque Conversion Coefficient a different approach was followed since a measured torque curve of the FZ90 Rocket Motor was not available. To do this the evolution of speed of gas in subsonic boundary layer was assumed to be the same for both rocket motors.

This should be a reasonable assumption because they are similar in construction and size. Secondly, we assume that both rocket motors transfer similar quantities of rotational energy into their respective rocket bodies. For such a situation the torque N_{yy} necessary to result in angular acceleration $\frac{d^2\psi}{dt^2}$ of the rocket body is given by:

$$\begin{aligned} N_{yy} &= I_{yy}\frac{d^2\psi}{dt^2} \\ \Rightarrow \frac{d\psi}{dt} &= \frac{1}{I_{yy}}\int N_{yy}dt \end{aligned}$$

Equation 5

with I_{yy} the Moment of Inertia of the rocket body around the **Y**r axis. The rotational power P_r and energy E_r transferred into the rocket body is given by:

$$\begin{aligned} P_r &= N_{yy}\frac{d\psi}{dt} \\ \therefore E_r &= \int\frac{N_{yy}}{I_{yy}}\int N_{yy}dtdt \end{aligned}$$

Equation 6

Equation 6 was implemented in the DFBD of Figure 30 by assuming both the C17 and FZ90 rocket motors to have a typical value of $6{,}369.10^{-3}$ kgm^2 for I_{yy}. Through trial and error it was found that both motors will store similar amounts of rotational energies (2280 Joule) in their rocket bodies if the FZ90 Torque Conversion Coefficient has the value of 0,027 meter. Under these assumptions the evolution of torque generated by the C17 and FZ90 Motors at +22°C core temperature as simulated with the model of Figure 30 are as illustrated in Figure 32. It is evident that in the case of the FZ90 Rocket Motor the model predicts a torque of about 5 Nm that essentially remains constant over a 1,0 second period after which it decays exponentially. This is similar to the torque curve of the MK-66 MOD2 Rocket Motor as illustrated in Figure 28. Hence, until new insight on the mechanisms at work becomes available (see paragraph 8.19.1) the coefficients as determined with the test setup of Figure 30 will be assumed as correct.

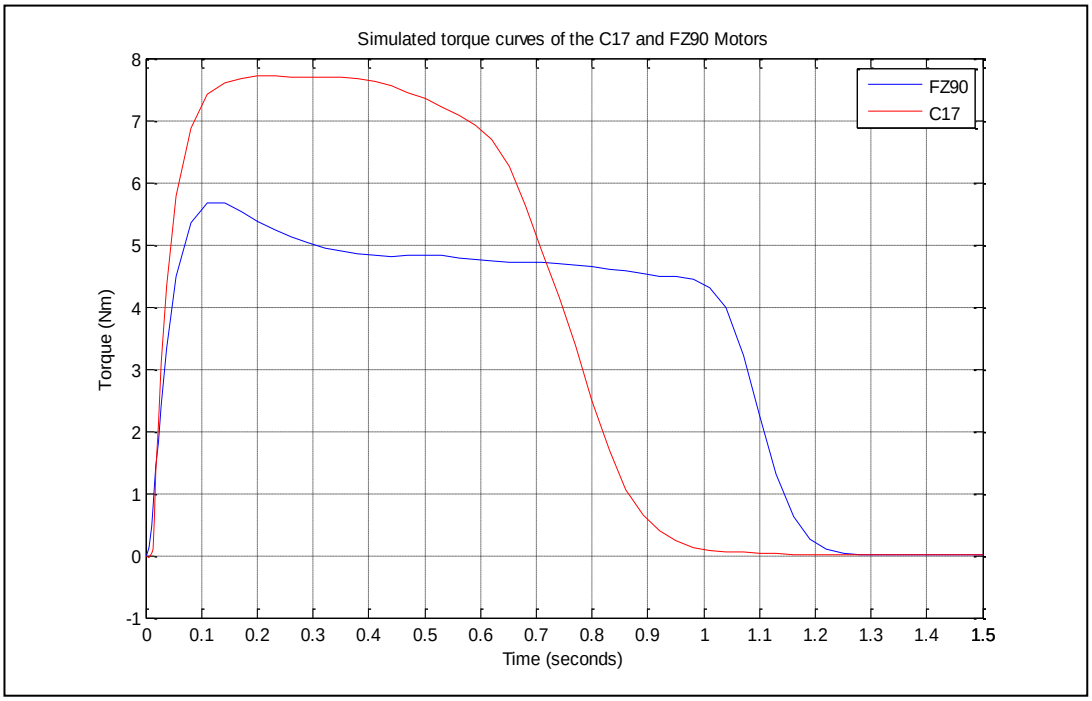


Figure 32. Simulated torque curves for C17 and FZ90 rocket motors [A Landman, 2014]

2.5.6.4 The FZ90 Centre of Pressure Estimator block

The **FZ90 Centre of Pressure Estimator** block calculates the Centre of Pressure (CoP) position of the FZ90 Rocket as function of Mach number. It does this with a lookup table containing a relationship as illustrated in Figure 33. This relationship was calculated with the AEROLAB program assuming the geometry of the FZ90 Rocket.

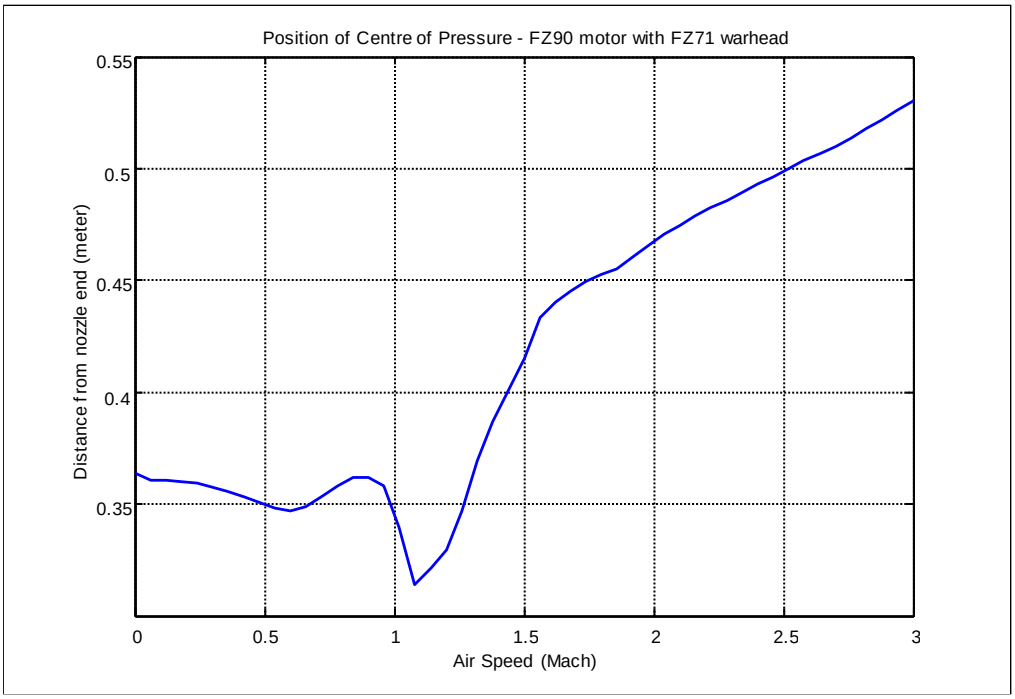


Figure 33. Centre of Pressure versus Mach number for FZ90 Rocket [A landman, 2014]

Note: *AEROLAB is a freeware CAD program written by Hans Olaf Toft. The program estimates (amongst others) Drag, Lift and Centre of Pressure for rockets flying at velocities in the range 0 to Mach 3 (extendable to Mach 8).*

Ideally, this information should be based on empirical data, but since suitable CoP data for the FZ90 Rocket was not available, it was based on an estimate as produced by the AEROLAB program. Although the source code of the AEROLAB program is not available, it would appear that the program uses a procedure similar to that developed by *Barrowman* [43] who used a collection of equations for calculating a wide range of aerodynamic characteristics of slender finned vehicles (such as the FZ90 Rocket) at small angles of attack. *Barrowman* [43] derives a set of equations by combining equations already derived by other authors in the aerodynamic field or by extracting relationships from empirical data. Testing the predictions of his equations for the Aerobee 350 Sounding Rocket, Tomahawk Missile and Basic Finner (standard model used in wind tunnel work) against empirical data, *Barrowman* [43] found correspondence within 10%.

2.5.6.5 The FZ90 Physical Characteristics Estimator block

The **FZ90 Physical Characteristics Estimator** block estimates the evolution in mass, Centre of Mass and Moments of Inertia of the FZ90 Rocket. These parameters change during the launch phase of the rocket (first 1,17 sec of flight) after which they remain constant. The DFBD of the **FZ90 Physical Characteristics Estimator** block is illustrated in Figure 34. No concise model comparable to the **FZ90 Physical Characteristics Estimator** block could be found in the open literature.

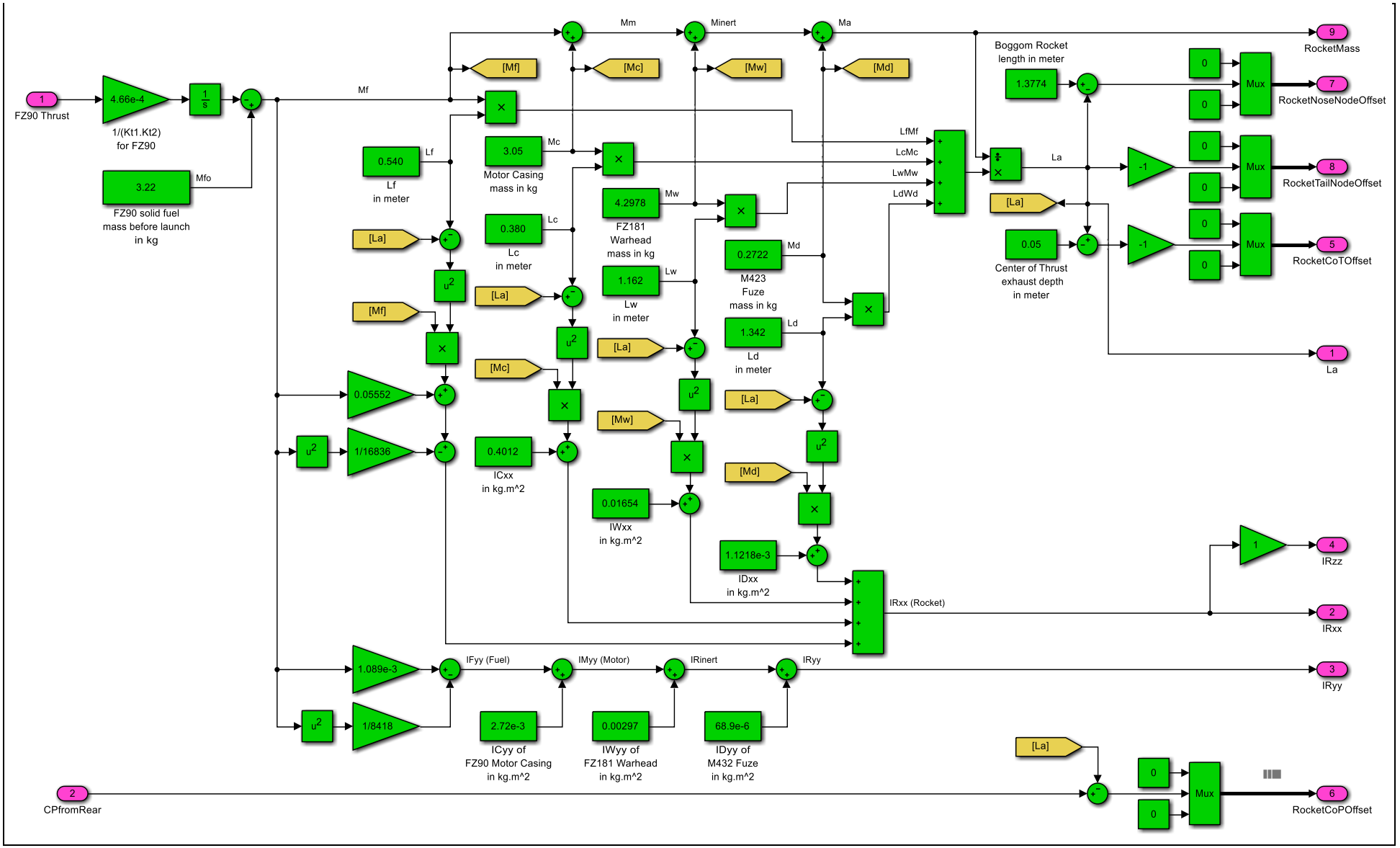


Figure 34. DFBD for FZ90 Physical Characteristics Estimator block [A Landman, 2014]

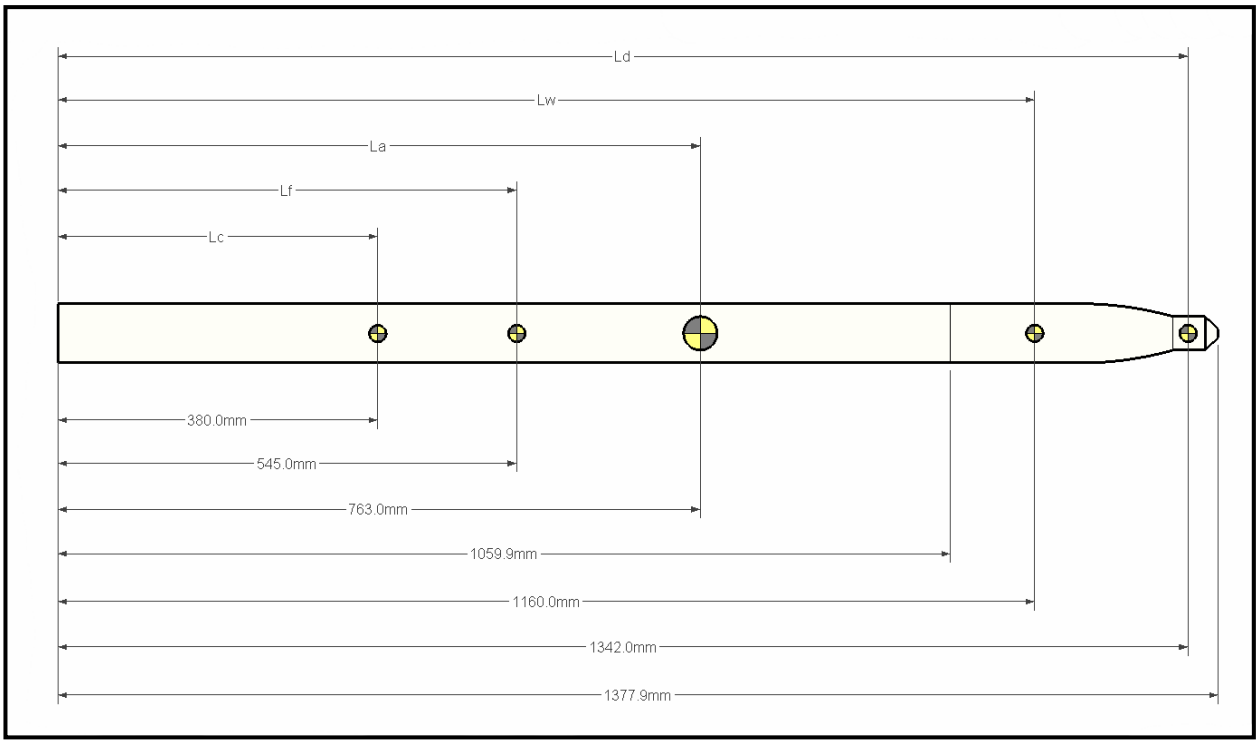


Figure 35. CoM positions for FZ90 Rocket before launch [A Landman, 2014]

To estimate the CoM and MoI of the FZ90 Rocket the rocket body is divided into multiple components – Solid Fuel, Casing, Warhead and Fuze as illustrated in Figure 35. This division is necessary because the Solid Fuel is consumed during launch and different Warheads and Fuzes have different masses and CoP positions

Given the assembly of rocket components as illustrated in Figure 35, we note that the following relationship must hold:

$$\begin{aligned} L_f M_f + L_c M_c + L_w M_w + L_d M_d &= L_a M_a \\ \Rightarrow \quad L_a &= \frac{L_f M_f + L_c M_c + L_w M_w + L_d M_d}{M_a} \\ &\equiv \frac{L_f M_f + L_c M_c + L_w M_w + L_d M_d}{\left(M_f + M_c + M_w + M_d\right)} \end{aligned}$$

Equation 7

with L_a the CoM position of the whole assembly, L_f the CoM position of the Fuel, L_c the CoM position of the Casing, L_w the CoM position of the Warhead and L_d the CoM position of the Fuze or detonator. M_a is the mass of the whole assembly, M_f is the mass of the Fuel, M_c is the mass of the Casing, M_w is the mass of the Warhead and M_d is the mass of the Fuze. Note that the value of L_a increases and the value of M_f decreases during launch as the solid fuel is consumed while the other parameters of Equation 7 remain constant.

2.5.6.5.1 Mass of Solid Fuel

To determine the evolution of the solid fuel mass of the FZ90 Rocket Motor, we note that as a first approximation the thrust of a rocket motor is directly proportional to the difference between ambient pressure and internal pressure of the combustion chamber:

$$F_t = k_e\left(P_i - P_a\right)$$

Equation 8

where F_t is the magnitude of the thrust vector, P_i is the internal pressure of the combustion chamber, P_a is the ambient pressure and k_e is an efficiency factor determined by the nozzle geometry. From the Bernoulli equation, we also know that the pressure differential over a venturi is directly proportional to the mass flow through it, so that we can write for the pressure differential over the nozzle:

$$\left(P_i - P_a\right) = -k_g \dot{m}_t \qquad \textbf{Equation 9}$$

where $\dot{m}_t$ is mass flow rate of fluid passing through the nozzle and k_g is a constant determined by the geometry of the nozzle and the composition of the exhaust gas. Substituting Equation 8 into Equation 9 and evaluating the integral we find:

$$F_t = -k_e k_g \dot{m}_t$$
$$\Rightarrow M_f = M_{fo} - \frac{1}{k_e k_g} \int_0^t F_t dt \qquad \textbf{Equation 10}$$

where M_f is the mass of the solid fuel remaining after a burn time of t seconds. M_{fo} is the mass of the solid fuel at ignition and F_t is the size of the thrust vector. The $k_e k_g$ constant can be determined because M_f must become zero with the integral evaluated over the burn time of the motor (0,9 sec for the C17 Motor at +22°C and 1,17 second for the FZ90 Motor at +20°C). Equation 10 therefore provides a link between the mass of remaining solid fuel in a rocket motor and its associated thrust curve. Given the remaining mass of solid fuel, the evolution of CoM position and Moments of Inertia of the rocket during launch can be calculated.

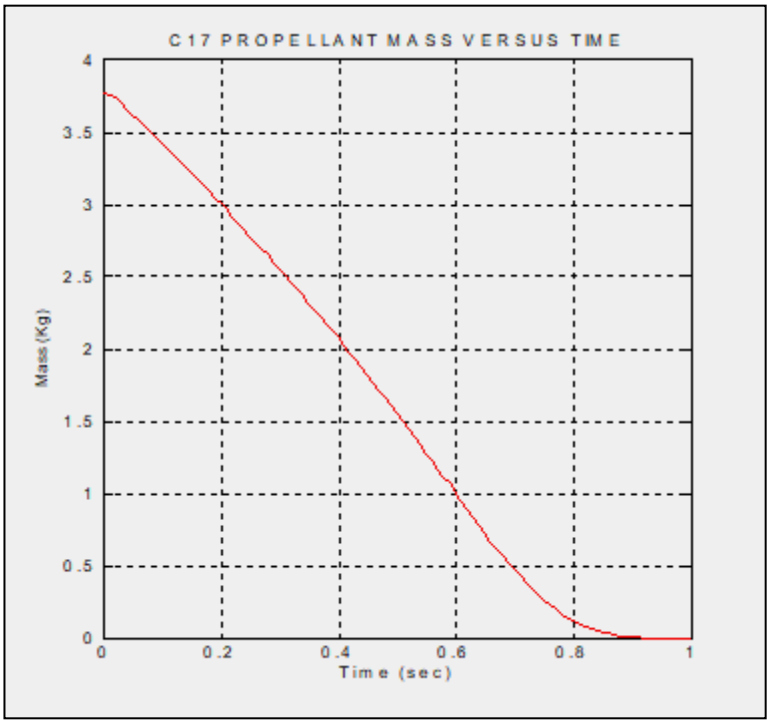


Figure 36. Depletion of propellant during C17 motor burn [A Landman, 2014]

The value of M_{fo} for the C17 Rocket Motor equals 3,81 kg[2]. Substituting this value for M_{fo} and the C17 thrust curve for F_t in Equation 10 and repeatedly evaluating the equation results in a

[2] As per Author data

value of 4,4366.10^{-4} for the $\frac{1}{k_e k_g}$ term to yield $M_f = 0$ at t = 0,9 sec. Hence, the depletion of solid propellant in the C17 Rocket Motor during launch at +22°C must be as illustrated in Figure 36. It is clear from this figure that just about all the propellant has been depleted after 0,8 seconds.

In similar fashion, M_{fo} for the FZ90 Motor equals 3,22 kg[3]. Applying the same procedure to the FZ90 Rocket Motor yields a value of 4,66.10^{-4} for the $\frac{1}{k_e k_g}$ term.

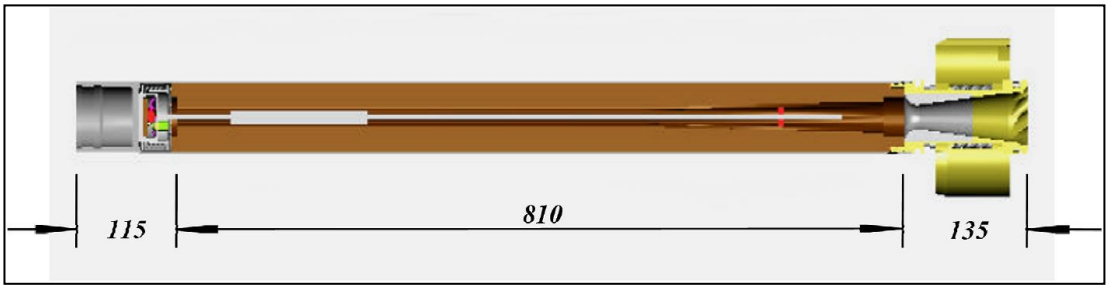


Figure 37. Primary dimensions of the M66 Hydra Motor [A Landman, 2014]

To determine the CoM of the Solid Fuel, we note that it is essensially a pipe which burns outwards from the entire inside surface, so the mass of the Solid Fuel will become less while its CoM will remain at the same position. Hence, L_f can be approximated as a constant. To determine the value of L_f refer to Figure 37 which shows a cross-sectional view of the MK-66 Rocket Motor which is very similar to the FZ90 Rocket Motor. From this figure it follows that the CoM of the Solid Fuel must be very close to the geometric centre of the brown segment, so parameter L_f must have a value of 0,540 meter for the FZ90 Rocket Motor.

2.5.6.5.2 Mass of other FZ90 components

To determine the mass and Centre of Mass of the other components of the FZ90 Rocket requires standard equations for the parallel axis theorem and MoI formulas as found in many physics handbook such as *Spiegel* [38] or *Kittel*, *Knight* and *Ruderman* [44]. The real problem here is finding trustworthy data for the FZ90 Rocket and separating truths from half-truths. Ultimately, the data should be acquired through physical measurement.

Table 1. Mass and positions of FZ90 Rocket Motor components

Component	Position on Yr Axis (meter)		Mass (kg)	
Solid Fuel	L_f	0,540	M_f	**Equation 10**
Motor Casing	L_c	0,380	M_c	3,050
FZ71 HE Warhead	L_w	1,162	M_w	4,298
M423 Impact Fuze	L_d	1,342	M_d	0,272

For the purposes of this study the required information was collected from various sources such as data acquired by the Author from Forges de Zeebrugge, *Wonnacot* [9], *Dahlke* and *Batiuk*

[3] As per Author's data

[21], *Kim* and *Walbridge* [10] as well as *Daniels* and *Hardy* [24]. The results of this search are summarized in Table 1.

2.5.6.5.3 MoI of Solid Fuel

To determine the evolution of Moments of Inertia of the FZ90 Rocket Motor, the density of the solid fuel must be known. According to Author information the solid fuel used in the FZ90 Motor is **SD1171**. An extensive search yielded no information on this material, so its density had to be determined through calculation.

To do this, the radius of the drill-hole of the FZ90 Rocket Motor was required. The only information on this parameter that could be found from the open literature was a presentation by *Hawley* [11] on the motivation for the K_2SO_4 rod modification of the M66 MOD4 Rocket Motor. It is evident from this presentation that the drill-hole radius of the M66 MOD4 Rocket Motor must be larger than 6,35 mm to accept the K_2SO_4 rod – scaling of the components in Figure 37 would suggest a drill-hole radius of about 8 mm. In addition, the drill-hole is not round but has a so-called 8 point burning star shape. Since the M66 Motor and FZ90 Motor are very similar in design and assuming the fins of the star to be about 20 mm deep, an effective drill-hole radius of about 18 mm is probably a reasonable assumption for the FZ90 Motor. Hence, the following relationship must hold:

$$\begin{aligned} \rho_{sf} &\approx \frac{M_{fo}}{\pi L_{rm}\left(R_2^2 - R_{bo}^2\right)} \\ &\equiv \frac{3{,}22}{\pi\, 0{,}810\left(0{,}033^2 - 0{,}018^2\right)} \\ &= 1654\, kg/m^3 \end{aligned} \qquad \textbf{Equation 11}$$

with ρ_{sf} the density of SD1171 solid fuel and L_{rm} the length of the Solid Fuel casting (from Figure 37 it is evident that parameter L_{rm} has a value of about 0,810 meter). Hence, the radius of the burn face for a given mass of solid fuel remaining in the motor is given by:

$$\begin{aligned} M_f &= \rho_{sf}\pi L_{rm}\left(R_2^2 - R_b^2\right) \\ \Rightarrow \left(R_2^2 - R_b^2\right) &= \frac{M_f}{\rho_{sf}\pi L_{rm}} \\ \Rightarrow R_b^2 &= R_2^2 - \frac{M_f}{\rho_{sf}\pi L_{rm}} \end{aligned} \qquad \textbf{Equation 12}$$

Substituting Equation 12 in the MoI formula of a thick-walled cylinder as described by *Wertz* and *Larson* [35] we have:

$$\begin{aligned} IF_{yy} &\approx \frac{M_f}{2}\left(R_2^2 + R_b^2\right) \\ &= \frac{M_f}{2}\left(R_2^2 + R_2^2 - \frac{M_f}{\rho_{sf}\pi L_{rm}}\right) \\ &= M_f R_2^2 - \frac{M_f^2}{2\rho_{sf}\pi L_{rm}} \\ &\equiv 1{,}089.10^{-3} M_f - \frac{M_f^2}{8418} \end{aligned} \qquad \textbf{Equation 13}$$

with IF_{yy} the MoI of the solid fuel around the **Y**r axis and with the origin of the Solid Fuel axes at the CoM of the Solid Fuel casting. To derive a relationship for the MoI of the solid fuel around the **X**r and **Z**r axes we apply the same formula and find:

$$\begin{aligned} IF_{xx} = IF_{zz} &\approx \frac{M_f}{12}\left[3\left(R_2^2 + R_2^2 - \frac{M_f}{\rho_{sf}\pi L_{rm}}\right) + L_{rm}^2\right] \\ &\equiv \frac{M_f\left(6R_2^2 + L_{rm}^2\right)}{12} - \frac{M_f^2}{4\rho_{sf}\pi L_{rm}} \\ &\equiv 0{,}05522M_f - \frac{M_f^2}{16836} \end{aligned} \qquad \textbf{Equation 14}$$

with IF_{xx} and IF_{zz} the MoI of the solid fuel around the Xr and Zr axes.

2.5.6.5.4 MoI of other FZ90 components

To determine the Moments of Inertia of the other components of the FZ90 Rocket Motor requires standard equations for the parallel axis theorem and MoI formulas as found in many physics handbooks such as *Spiegel* [38] or *Kittel*, *Knight* and *Ruderman* [44]. Again the real problem is finding trustworthy data for the FZ90 Rocket and separating the truth from half-truths. Ultimately, the data should be acquired through physical measurement.

For the purposes of this study the required information was collected from various sources such as Authors own partial data on the FZ90 rocket family, *Wonnacot* [9], *Dahlke* and *Batiuk* [21], *Kim* and *Walbridge* [10] as well as *Daniels* and *Hardy* [24]. The results of this search are summarized in Table 2.

Table 2. MoI of FZ90 Rocket Motor components

Component	MoI around Xr Axis (kg.m^2)		MoI around Yr Axis (kg.m^2)		MoI around Zr Axis (kg.m^2)	
Solid Fuel	IF_{xx}	**Equation 14**	IF_{yy}	**Equation 13**	IF_{zz}	**Equation 14**
Motor Casing	IC_{xx}	0,4012	IC_{yy}	0,0027	IC_{zz}	0,4012
FZ71 HE Warhead	IW_{xx}	0,0165	IW_{yy}	0,0030	IW_{zz}	0,0165
M423 Impact Fuze	ID_{xx}	$1{,}122.10^{-3}$	ID_{yy}	$6{,}890.10^{-5}$	ID_{zz}	$1{,}122.10^{-3}$

2.5.6.5.5 Testing the FZ90 Physical Characteristics Estimator block

The evolution of Thrust and Moments of Inertia of only the FZ90 Rocket Motor as simulated with the **FZ90 Physical Characteristics Estimator** block is illustrated in Figure 38. Note that this simulation was conducted with mass and MoI of Warhead and Fuze set to zero.

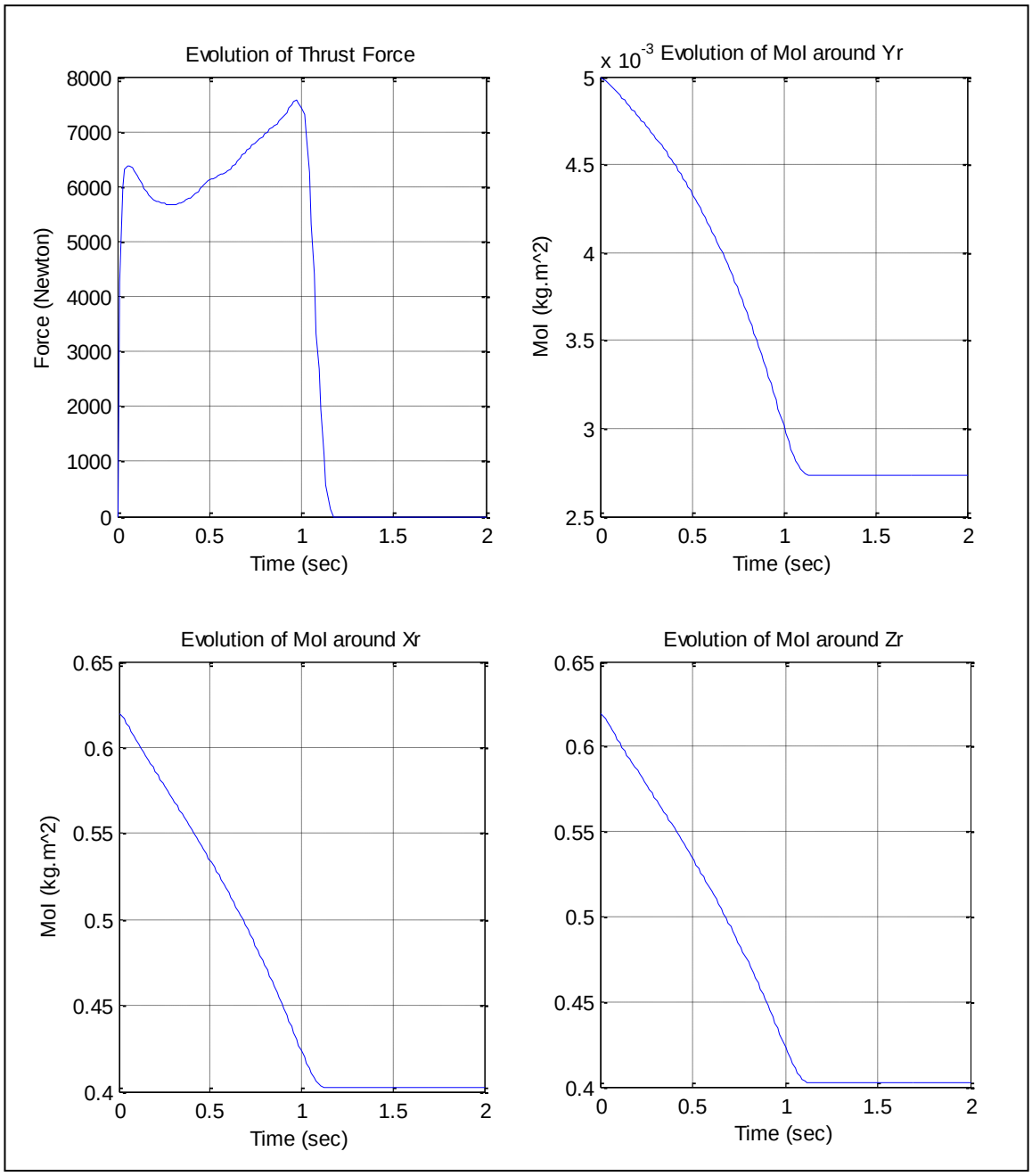


Figure 38. Simulated evolution of MoI: standalone FZ90 Rocket Motor [A Landman, 2014]

Table 3. Moments of Inertia: MK-66 Motor versus FZ90 Motor (simulated)

	Live		Fired	
	Axial ($kg.m^2$)	**Transversal ($kg.m^2$)**	**Axial ($kg.m^2$)**	**Transversal ($kg.m^2$)**
MK-66 Motor	0,0046	0,5946	0,0027	0,4012
FZ90 Motor	0,0050	0,6190	0,0027	0,4000

Comparing the results as illustrated in Figure 38 with the characteristics of the MK-66 MOD2 Rocket Motor as described by *Dahlke* and *Batiuk* [21] we find the values as listed in Table 3.

For conditions before launch, the Axial Moments of Inertia are within 8% and the transversal Moments of Inertia within 4%. For conditions after launch the values are almost the same.

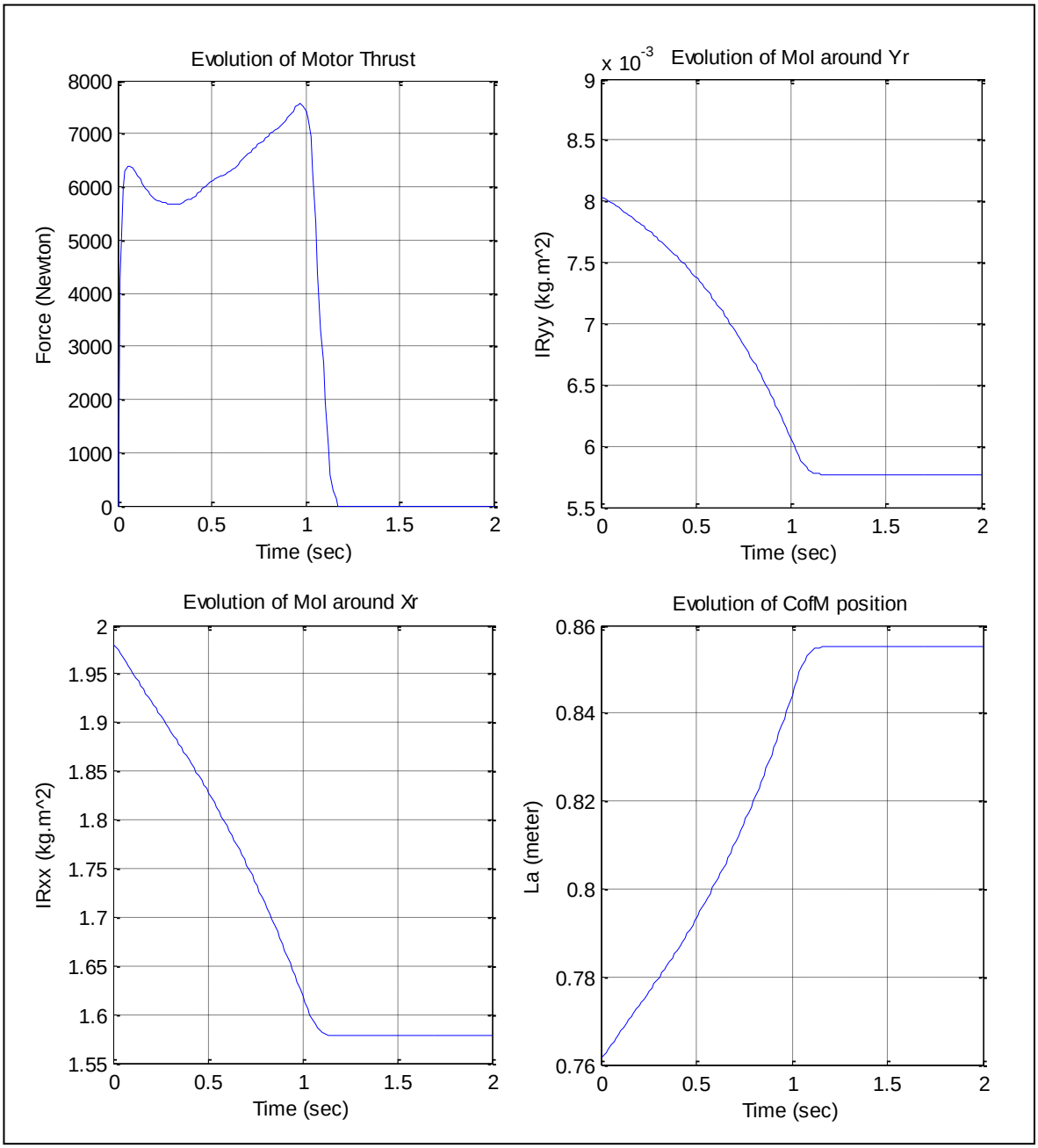


Figure 39. Simulated evolution of Thrust, MoI and CoM position of Boggom Rocket.

Table 4. Moments of Inertia: MK-66/M151 Rocket versus FZ90/FZ71 Rocket (simulated)

	Live		Fired	
	Axial ($kg.m^2$)	**Transversal ($kg.m^2$)**	**Axial ($kg.m^2$)**	**Transversal ($kg.m^2$)**
MK-66 & M151	0,0077	1,8284	0,0058	1,4655
FZ90 & FZ71	0,0080	1,9750	0,0057	1,5800

Similarly, the evolution of Thrust and Moments of Inertia of the FZ90/FZ71/M423 Rocket combination as simulated with the **FZ90 Physical Characteristics Estimator** block is illustrated in Figure 39. Comparing the results as illustrated in Figure 39 with the characteristics of the MK-66/M151 Rocket combination as described by *Dahlke* and *Batiuk* [21], we find the values as listed in Table 3. For conditions before launch the Axial Moments of Inertia are within 3,75% and the transversal Moments of Inertia within 7,4%. For conditions after launch the Axial Moments of Inertia are within 1,75% and the transversal Moments of Inertia within 7,3%. In addition, the **FZ90 Physical Characteristics Estimator** block indicates a 93 mm shift in Centre of Mass versus 91,1 as as described by *Dahlke* and *Batiuk* [21]. Considering the uncertaincies and variations in mass between FZ90 and MK-66 rocket families these differences are small, suggesting that the **FZ90 Physical Characteristics Estimator** block operates correctly.

Note: *No data describing the evolution of mass and MoI of the C17, FZ90 or MK-66 rockets from state Live to state Fired could be found in the open literature.*

The test as described above demonstrates one of the problems to be mitigated by the PRLS - unless a given characteristic is physically measured there can be no guarantee as to the accuracy of the simulation, no matter how good the model. Similar components made in different batches or by different manufacturers result in small differences which can affect the accuracy of the prediction. This is one of the error mechanisms over which snipers exercise strict control (see paragraph 1.2.2).

Note: *The same firing table cannot serve the purpose of all Rocket Configurations as currently done on Rooivalk AH-2A. Each Rocket Configuration must have a separate firing table, and the Aircraft-Store Interface (ASI) must make provision for telling the system what Rocket Configuration is currently loaded.*

2.5.7 The Rocket-Launcher I/O block

This block forms part of the **Rocket Launching System Model** as illustrated in Figure 20. The DFBD for the **Rocket-Launcher I/O** block is illustrated in Figure 40. The objective of this block is to provide a real time model of the interface (i.e. Rattle Fit) between FZ90 Rocket and M159 Launcher. Determining the interaction between FZ90 Rocket and M159 Rocket Launcher during launch is necessary because it determines the initial conditions of the FZ90 Rocket as it enters the downwash. In particular, movement of the M159 Rocket Launcher during launch can lead to mallaunch conditions (tip-off) as described in *MIL-HDBK-762 (MI)* [29].

No model capable of simulating the Rattle Fit Interface in real time and with reasonable accuracy could be found in the open literature. *Lee*, *Choi* and *Kwon* [28] used a finite element and oversets method to calculate the interactions between main rotor, tail rotor and fuselage of a UH-60 helicopter in conjunction with a rocket with fins and canards. They model movement of the rocket in the launch tube as single DOF movement (i.e. rocket movement only allowed along launcher centre line). *Wonnacott* [9] overcomes the problem by initializing his 6 DOF model with different sets of flight conditions, most of which describe an ideal state of the MK-66 Rocket some time into the launch sequence. Paragraph 4-45 of *MIL-HDBK-762 (MI)* [29] contains a rudimentary description of some of the interactions between rocket and launcher that can lead to mallaunch conditions, but provides no specific model to simulate these effects. A dedicated model capable of performing this function was therefore developed under this study.

Note: *The **Rocket-Launcher I/O** block does not model the dynamic behaviour of the FZ90 Rocket nor the M159 Launcher. It models the behaviour of Space between the FZ90 Rocket and M159 Launcher given the positions and rotations of these rigid bodies. Space therefore reacts like a type of kinematic joint in this model.*

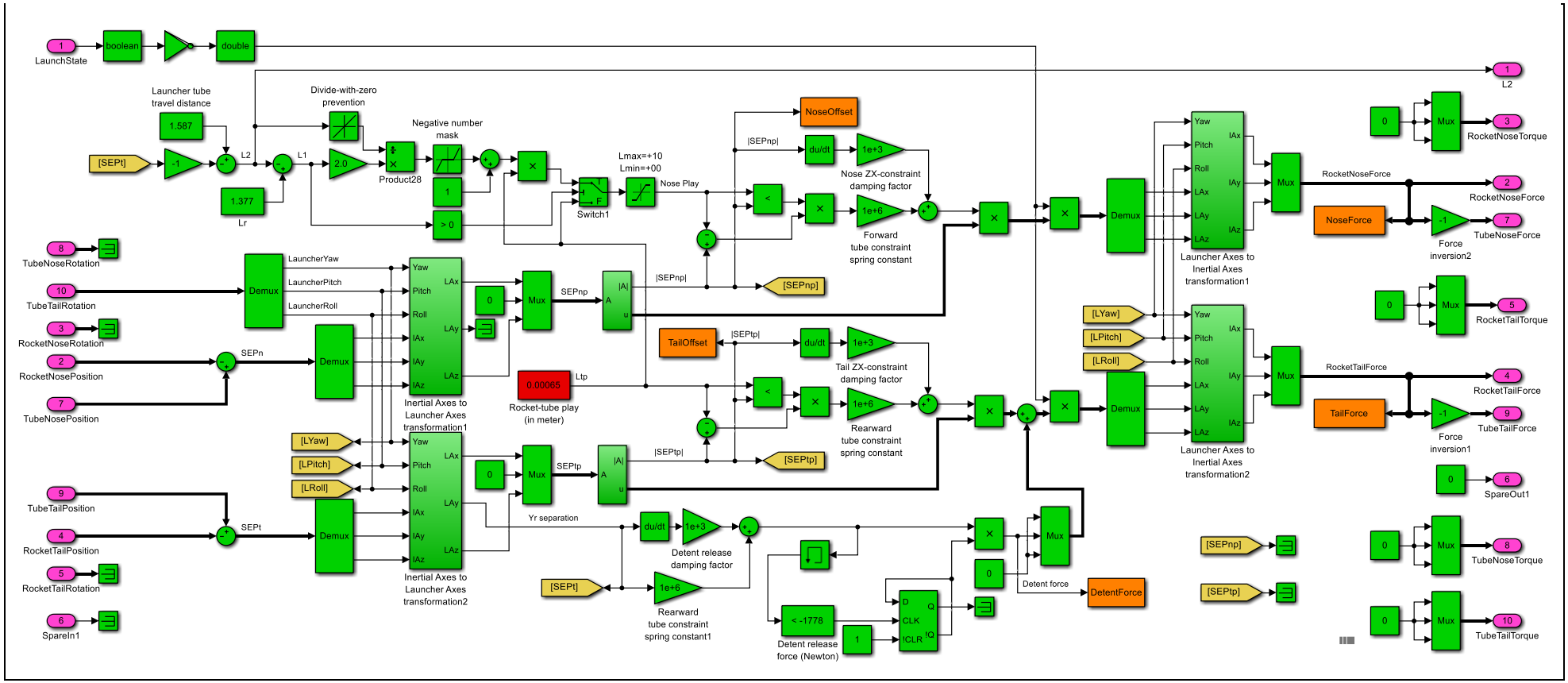


Figure 40. DFBD of the Rocket-Launcher I/O block [A Landman, 2014]

The approach followed to calculate the response of the minute region of Space located between FZ90 Rocket and M159 Rocket Launcher is illustrated in Figure 41. In this diagram both the FZ90 Rocket and M159 Rocket Launcher are assumed to be rigid bodies, each with a node located at the front (nose) and a node located at the rear (tail). In such an arrangement the M159 Launcher imposes an envelope of constraints on the position and orientation that the FZ90 Rocket can assume. The "constraint envelope" as illustrated by the dotted lines in Figure 41 designates the region of allowable positions for the nose and tail nodes of the FZ90 Rocket from the moment of ignition until the rocket leaves the Launch Tube. The evolution of orientations and positions that both the FZ90 Rocket and M159 Rocket Launcher assume during launch thus depend on the dynamics of and collisions between the FZ90 Rocket and M159 Launcher.

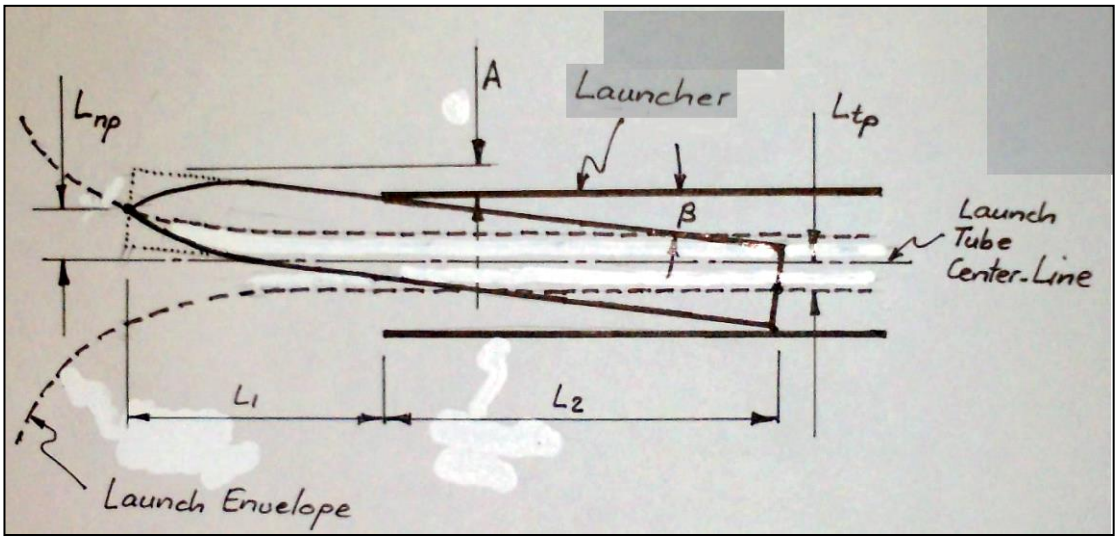


Figure 41. Graphical representation of Rattle Fit Interface [A Landman, 2014]

To derive an expression for the constraint envelope we have from Figure 41 for β small:

$$\begin{aligned} \frac{L_a}{L_1} &= \frac{2L_{tp}}{L_2} \\ \Rightarrow \quad L_a &= \frac{2L_{tp}L_1}{L_2} \\ &\approx \frac{2L_{tp}\left(L_r - L_2\right)}{L_2} \end{aligned} \qquad \textbf{Equation 15}$$

with the symbols as defined in Figure 41. Note that during the entire travel of the Rocket through the Launch Tube the play of the Rocket Tail Node is simply equal to L_{tp}. To derive an equation for the play L_{np} of the Rocket Nose Node we have from Figure 41 and Equation 9:

$$\begin{aligned} L_{np} &= \frac{D_t}{2} + L_a - \frac{D_r}{2} \\ &\equiv \frac{\left(D_r + 2L_{tp}\right)}{2} + \frac{2L_{tp}\left(L_r - L_2\right)}{L_2} - \frac{D_r}{2} \\ &\equiv L_{tp}\left[1 + \frac{2\left(L_r - L_2\right)}{L_2}\right] \qquad \textit{if } L_1 > 0 \textit{ else } L_{np} = L_{tp} \end{aligned} \qquad \textbf{Equation 16}$$

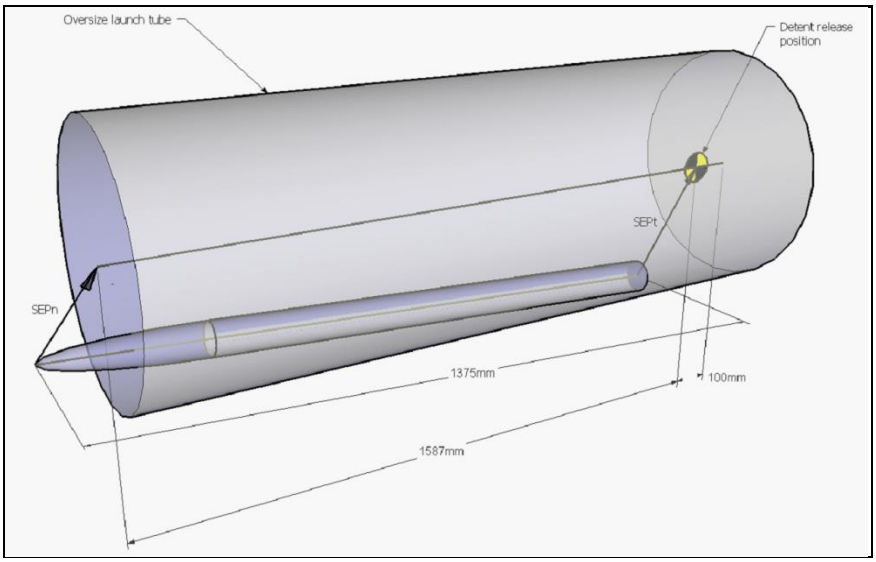


Figure 42. Separation vectors of Rocket in Launch Tube [A Landman, 2014]

To calculate the movements of the nose node and tail node of the FZ90 Rocket with respect to the constraint envelope, consider a Rocket located in an oversize Launch Tube with the Rocket displaced from the Launch Tube centre line both in translation and rotation as illustrated in Figure 42. Introduce connecting nodes on the FZ90 Rocket at the centres of its nose and tail. Designate these nodes as the Rocket Nose Node and Rocket Tail Node respectively. Similarly, introduce connecting nodes on the Launch Tube at its forward centre and Detent Release centre. Designate these nodes as the Tube Nose Node and Tube Tail Node respectively.

Introduce the vector ***SEP$_t$*** to designate the separation from Rocket Tail Node to Tube Tail Node and the vector ***SEP$_n$*** to designate the separation from Rocket Nose Node to Tube Nose Node. Also introduce unit vector ***ur*** along the Rocket center line, pointing from Rocket Tail Node to Rocket Nose Node and unit vector ***ut*** along the Launch Tube center line, pointing from Tube Tail Node to Tube Nose Node. For such a configuration, vector ***SEP$_n$*** can be described as follows:

$$\overline{SEP}_n = \overline{TNosePos} - \overline{RNosePos} \qquad \textbf{Equation 17}$$

with ***TNosePos*** and ***RNosePos*** the position vectors of the nose nodes of the tube and rocket respectively. Similarly, vector ***SEP$_t$*** can be described as follows:

$$\overline{SEP}_t = \overline{TTailPos} - \overline{RTailPos} \qquad \textbf{Equation 18}$$

with ***TTailPos*** and ***RTailPos*** the position vectors of the tail nodes of the tube and rocket respectively.

With the separation vectors and the shape of the constraint envelope known, the forces acting on the FZ90 Rocket due to excursions of the nose node and tail node of the FZ90 Rocket can now be calculated. The Rocket-Launcher Interface was assumed to react to these excursions like a very stiff spring with a dead band corresponding with the constraint envelope. Experimentation with the **Rocket-Launcher I/O** block showed that a spring constant of 1.10^{+6} Newton/meter gave a very reasonable result without incurring a large computation time penalty. In similar fashion, the layer of air trapped between FZ90 Rocket and Launch Tube was assumed to react like a shock absorber which causes a force opposite to lateral movements of the FZ90 Rocket within the Launch Tube. Experimentation with the **Rocket-Launcher I/O** showed that a damping factor of 1.10^{+3} Newton.sec /meter gave very reasonable results without incurring a large computation time penalty.

By translating the separation vectors into Local Launcher Axes and using the relationships as described above the forces acting on the FZ90 Rocket and M159 Launcher are calculated as illustrated in Figure 40. After suitable summation the resulting forces are translated back into Inertial Axes for use in the **Rocket Launching System Model** of Figure 20. Note that the Detent Release Mechanism is modelled with a D-type flip flop that releases the FZ90 Rocket when the thrust exceeds 1778 Newton[4].

2.5.7.1.1 Testing the Rocket-Launcher I/O block

The response (movement) of the FZ90 Rocket during installation in a M159 Launcher Tube was simulated by combining the **FZ90 Rocket Model** block of Figure 22 with the **Rocket-Launcher I/O** block of Figure 40. In this simulation, the position inputs normally generated by the **M159 Launcher** block (see Figure 20) was clamped at a Super Elevation angle of +30° while the state vector of the FZ90 Rocket was initialized with centre line of FZ90 Rocket close to centre-line of M159 Launcher and the tail of the FZ90 Rocket close to the Detent Release Mechanism. The evolution of the magnitudes of the separation vectors $\boldsymbol{SEP_t}$ and $\boldsymbol{SEP_n}$ (in Launcher Axes) and the forces acting on the tail and nose nodes of the Rocket (in Inertial Axes) as the FZ90 Rocket settles towards equilibrium is illustrated in Figure 43.

Note: *All partial builds of the **Rocket Launching System Model** must be "integrated" prior to running the actual simulation. This entails aligning the initial conditions of connecting nodes between adjacent blocks as close as possible and allowing the blocks comprising the system to settle into an equilibrium (i.e. integrated) position. This is necessary to avoid excessive separations in the kinematic joints which tend to cause the model to crash.*

It is evident from Figure 43 that the FZ90 Rocket was released very close to the Launcher center line and that it starts falling towards the bottom of the Launch Tube. After making contact with the bottom of the Launch Tube, it settles at a displacement that represents excursions of about 45 micron at both its nose and tail nodes. At equilibrium these excursions should be such that the sum of the forces acting on the FZ90 Rocket is zero. That this is indeed so along the Up-axis is evident from Figure 43 because the sum of the upward force on the Rocket Nose Node (42,9 Newton) and the Rocket Tail Node (60,6 Newton) equals the weight of the Boggom Rocket (103,5 Newton downwards through CoM). Along the North-axis the forces acting on the Rocket Nose Node and the Rocket Tail Node are both 25 Newton but opposite in direction.

Simulating the movement of the FZ90 Rocket when the Rocket is fired after allowing it to settle in the Launcher Tube under the conditions as described above yields the response illustrated in Figure 44. From this diagram, it is evident that the Rocket Tail Node remains close to the lower edge of the launch envelope (i.e. it slides along the bottom of the Launch Tube) as expected until Launcher Exit occurs. In contrast, the Rocket Nose Node starts to fall when the nose of the Rocket begins to leave the Launcher, ending at an offset of 2,58 mm below the launch tube centre line when Launcher Exit occurs. This is to be expected, since the speed of the FZ90 Rocket under these conditions is relatively low, allowing enough time for gravity to accelerate the FZ90 Rocket downwards. The evolution in attitude of the FZ90 Rocket under these conditions is illustrated in Figure 45 from which it is evident that the drooping of the Boggom Rocket under this particular set of launch conditions represents an error of 0,08° in pitch at Launcher Exit. Yawing of the FZ90 Rocket inside the Launch Tube is observed along the yaw axis as the FZ90 Rocket tries to conserve its momentum when its nose starts to drop when it exits the Launch Tube.

[4] Design choice made by Forges De Zeebrugge

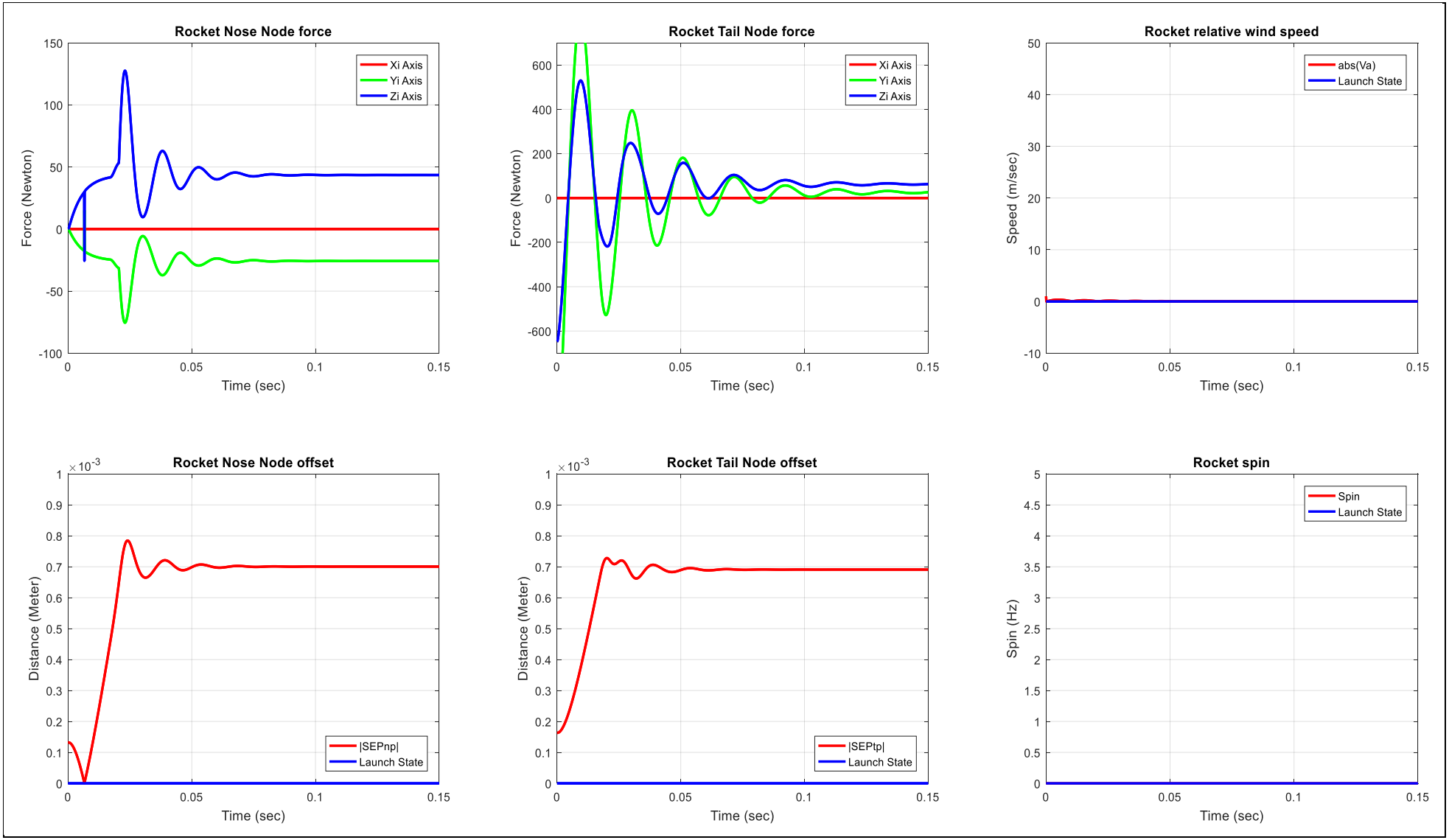


Figure 43. Movement of FZ90 Rocket during rocket installation [A Landman, 2014]

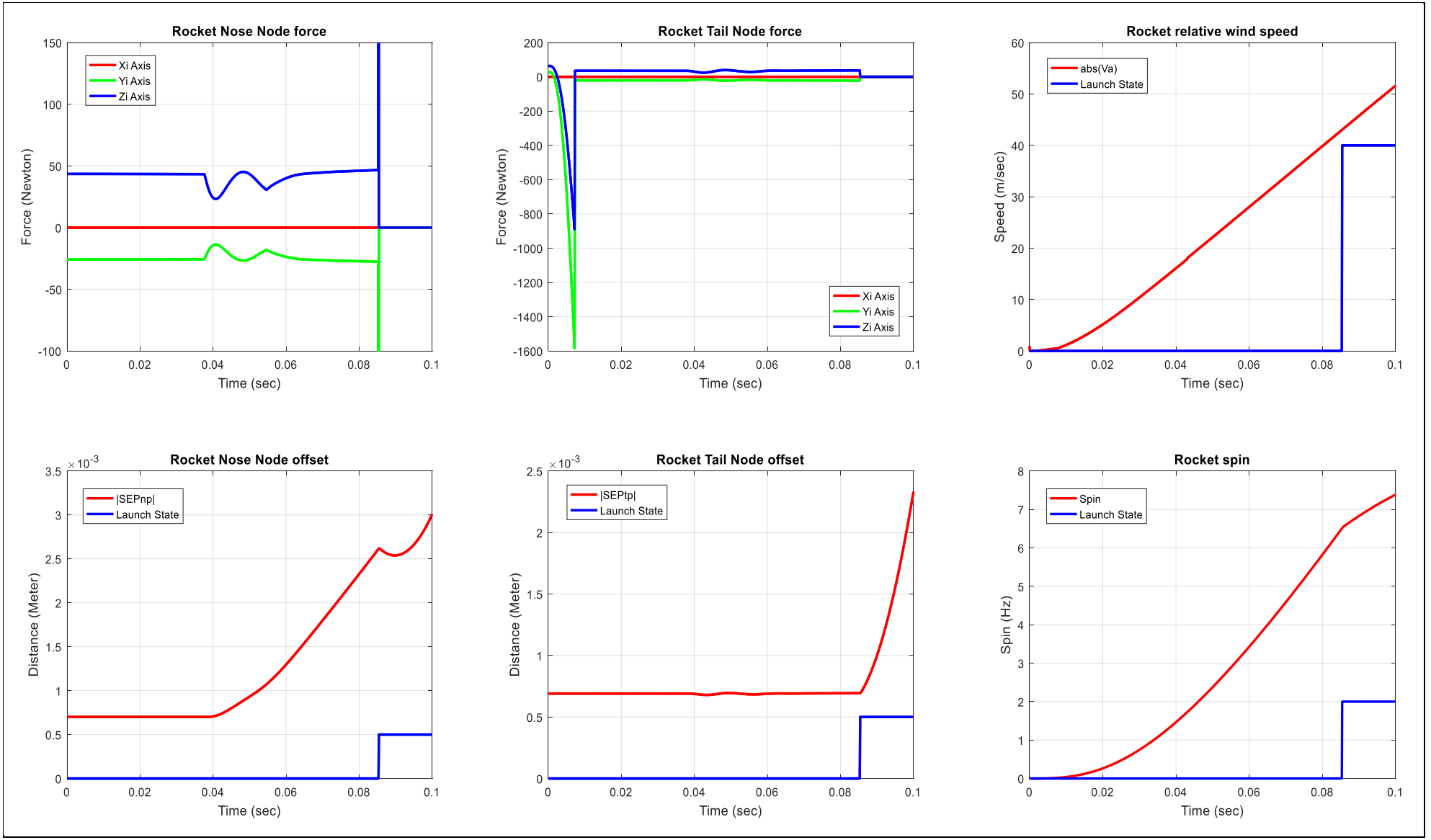


Figure 44. Movement of FZ90 Rocket while tranversing the Launch Tube [A Landman, 2014]

The force exerted on the Rocket Tail Node during launch is illustrated in the top centre graph of Figure 44. The effect of the Detent Release Mechanism can be clearly seen from this diagram, yielding a resultant force that builds up to a maximum over a period of 8 msec during which the Rocket remains stationary. When the resultant force reaches $\sqrt{856^2 + 1562^2} = 1781\,N$ the Detent Release Mechanism unlocks and the Rocket becomes free to move along the Launch Tube. Note that this is quite close to the 1778 Newton release threshold of the Detent Release Mechanism as discussed earlier. The time from rising edge of the firing pulse until Launcher Exit is 85 msec. During this period the forces acting on the Rocket Tail Node are those required to keep the node within the constraint envelope of the Rocket-Launcher Interface. Finally, we note from Figure 44 that the speed and spin rate of the Rocket at Launcher Exit are 43 m/s and 6,5 Hz respectively.

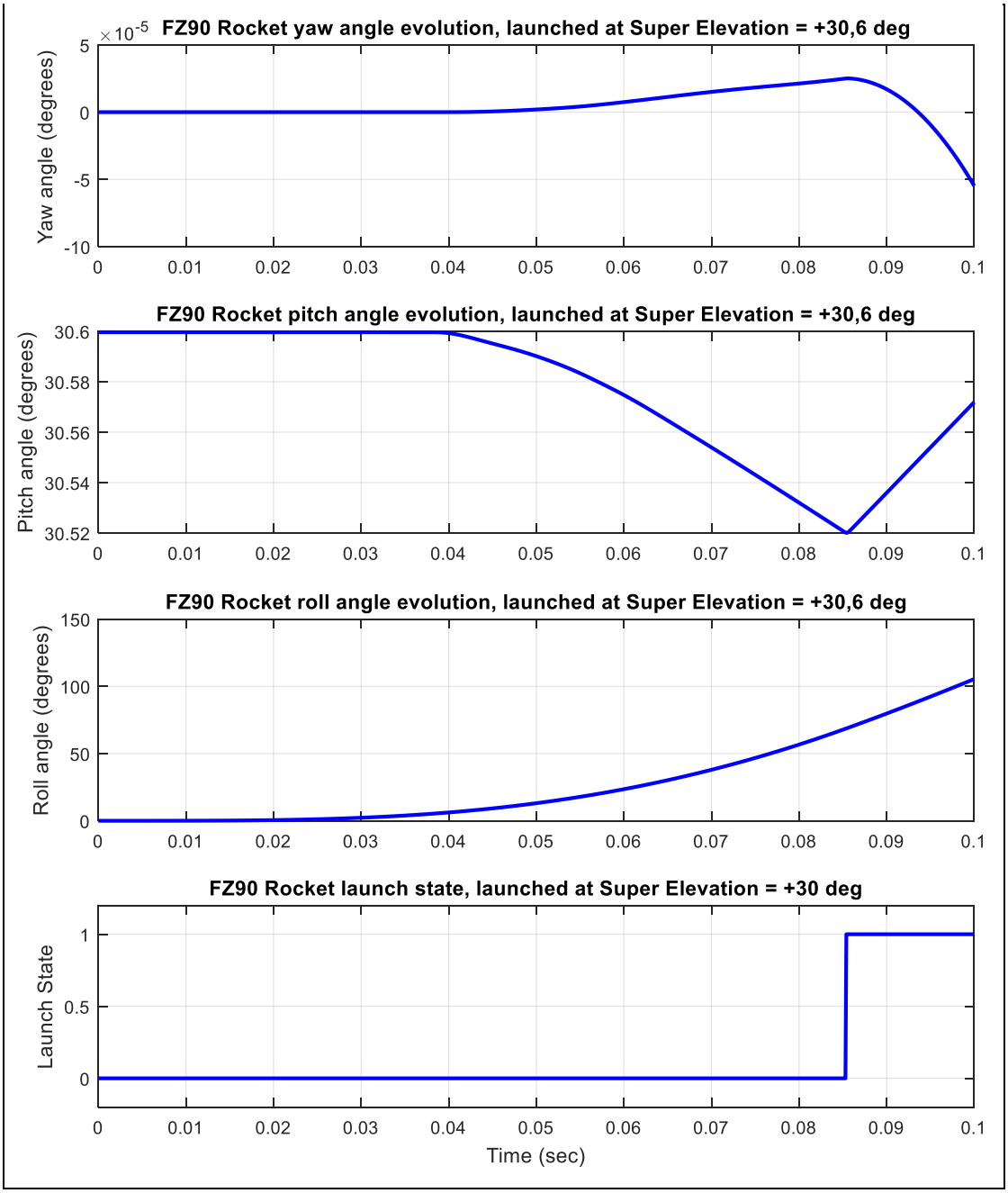


Figure 45. Evolution of FZ90 Rocket attitude: launched at SE = +30,6° [A Landman, 2014]

2.5.8 The Space block

The **Space** block forms part of the **Rocket Launching System Model** with data flows as illustrated in Figure 20. The diagram reflects the concept that Space is a fluid body that surrounds every rigid body of the Rocket Launching System. In such a model downwash is the response of Space due to to the movements (represented by ***Shape*** signals) of things such as the Main Rotor, Tail Rotor, Fuselage, Ground and even the FZ90 Rocket itself. Space reacts to these movements by generating various forces (represented by ***Press*** signals) acting on all of these rigid bodies. Note that some of these rigid bodies can also impose specific conditions on the boundary of Space, such as exhaust flows from the Engines of the Helicopter. This is the same approach as followed by *Lee*, *Choi* and *Kwon* [28] in modelling the exhaust plume of a rocket when they assume gasflow with velocity Mach one (i.e. a choked nozzle) and temperature 1600 °C over the entire exit area of the rocket nozzle. In terms of electrical circuit theory, they model the rocket exhaust as a current source. As shown by *Anderson* [45] all aerodynamic effects such as lift and drag can all be modelled in terms of this philosophy. For example, movement of a wing through air (Space) causes the air to respond by creating a pressure distribution over the surface of the wing. In turn, the wing integrates this pressure distribution to yield an overall force known as lift.

The philosophy as described above was also persued because it allows formulation of the **Rocket Launching System Model** in terms of multi-ports as described by *Karnopp*, *Margolis* and *Rosenberg* [31]. This philosophy is followed throughout this study, albeit with bulk equations for lift and drag in this version of the **Space** block.

Note: *In later chapters of this study the principle will be persued in much greater detail, resulting in the **Space** block containing a model with millions of DOF.*

For the purposes of the simulation required under this chapter, the response of Space is addressed as five separate aspects – rocket drag, rocket lift, fin torque, near field flow pattern (downwash), far field flow pattern (wind, gusts and atmospheric conditions). Combining these aspects results in the DFBD of the **Space** block as illustrated in Figure 46.

2.5.8.1 Drag of the FZ90 Rocket

Anderson [45] describes the drag acting on a body such as the FZ90 Rocket as comprising a basic drag component and an induced drag component which can be formulated as follows:

$$F_D = \frac{\rho_y V_a^2 C_{Do} A_o}{2} + \frac{\rho_y V_a^2 C_{Di} A_r}{2} \qquad \textbf{Equation 19}$$

with F_D the total drag force acting on the rocket in Inertial Axes, C_{Do} the zero-yaw drag coefficient of the rocket, C_{Di} the induced drag coefficient of the rocket, ρ_y the density of the air in the vicinity of the rocket, V_a the relative air velocity of the FZ90 Rocket in Inertial Axes. A_o represents the frontal surface area of the FZ90 Rocket in and A_r represents the lateral surface area of the rocket. Note that the drag force acts through the Centre of Pressure of the rocket and anti-parallel to $\bar{V}_a$.

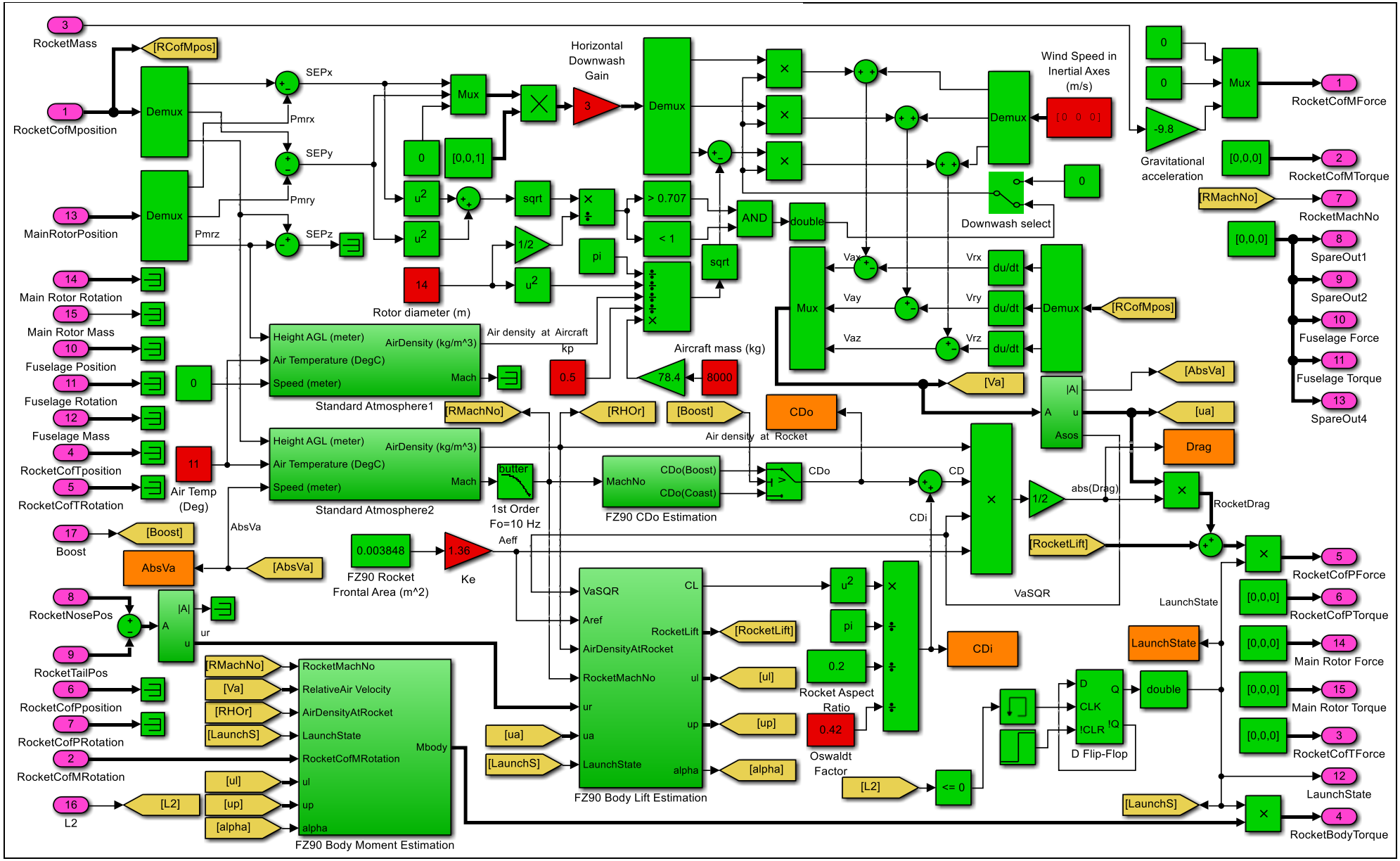


Figure 46. DFBD of the Space block [A Landman, 2014]

As described by *Anderson* [45] basic drag of the FZ90 Rocket is associated with the effects of surface shear and the effort needed to displace the air in front of the FZ90 Rocket (Space responds to this movement by creating a pressure distribution that opposes the movement). This drag is at minimum when the centre line of the FZ90 Rocket is aligned with its velocity vector with respect to the surrounding air and is characterized by the zero-yaw drag coefficient C_{Do}. The zero-yaw drag coefficient of the FZ90 Rocket is dependant on Mach number and is calculated as described in paragraph 2.5.8.1.1 for the purposes of this study.

As described by *Anderson* [45] induced drag of the FZ90 Rocket is caused when the centre line of the FZ90 Rocket is not aligned with its velocity vector with respect to Space. Under such conditions, Space reacts by generating a pressure and shear distribution which, when summed over the FZ90 Rocket Body, yields an induced drag force in accordance with Equation 19 and a lift force with characteristics as described in paragraph 2.5.8.2. Lift combined with movement implies that work is being done and the process conducting this work must have some measure of inefficiency, which finds expression in the form of an increase in drag force acting on the FZ90 Rocket. Hence, there is a direct connection between lift and induced drag of the FZ90 Rocket. As described by Anderson [45] the induced drag coefficient C_{Di} is given by the following equation:

$$C_{Di} = \frac{C_L^2}{\pi e AR_r}$$ **Equation 20**

with C_L the lift coefficient of the FZ90 Rocket, AR_r the aspect raio of the FZ90 Rocket and *e* the Oswald efficiency factor associated with the shape of the FZ90 Rocket. Typically *e* is equal to 1,0 for an elliptic wing and 0,7 for a rectangular wing. Since the FZ90 Rocket can be considered as a very inefficient wing, it has a very low Oswald efficiency factor. *Niţă* and *Scholz* [46] developed their own method to determine the Oswald efficiency factor after considering the methods developed by a (very) wide range of authors. Calculation of *e* is clearly not a simple task since it depends on various factors such as aspect ratio, twist and sweep of the wing. It is also dependant on Mach number. Worst of all, the methods are intended for a wing (i.e. a flat body with reasonable width), not the long slender body of a FZ90 Rocket. This parameter should therefore really be determined through wind tunnel tests. In the case of the FZ90 Rocket such data could not be found in the open literature, so the approach followed in this study is to construct the **Space** block using a theoretical estimate of the Oswald efficiency factor, suitably adapted through available flight test results. The Oswald efficiency factor for a wing with leading edge sweep angle Λ_{LE} greater than 30° is given by:

$$e = 4{,}61\,(1 - 0{,}045 AR^{0.68})(\cos \Lambda_{LE})^{0.15} - 3{,}1$$ **Equation 21**

Note that the approach taken by this study is to assume Λ_{LE} equal to the ogive angle of the FZ90 Rocket (about 80°). The aspect ratio AR_r of the rocket is given by the following equation:

$$AR_r = \frac{S_r^2}{A_r}$$ **Equation 22**

with S_r the width (i.e. span in case of a wing) of the Rocket in meter and A_r the lateral surface area of the Rocket in m^2. In the case of the Boggom Rocket, S_r = 0,07 meter and A_r = 0,07*1,377 = 0,09639 m^2 so that AR_r = 0,0508. From Equation 21 it then follows that for the FZ90 Rocket the Oswald efficiency factor equals about 0,42.

Note: *It must be noted here that for Equation 20 to hold, the same reference surface area should be used for calculating both lift and induced drag (i.e. A_o and A_r in Equation 19 are the same). This fact must be taken into consideration (through scaling) when using the lift coefficient of the FZ90 Rocket, as predicted with the AEROLAB program which uses the frontal surface area of the rocket as reference area.*

2.5.8.1.1 Determining the zero-yaw drag coefficient

To derive an estimate of the zero-yaw drag coefficient of the FZ90 Rocket, the AEROLAB program was used to calculate CD_o for a rocket with the dimensions of the FZ90 Rocket as defined in paragraph 2.5.6.5. The resulting CD_o curve for the FZ90 Rocket over its operational speed range (Mach 0 to Mach 3) is illustrated by the blue curve of Figure 47.

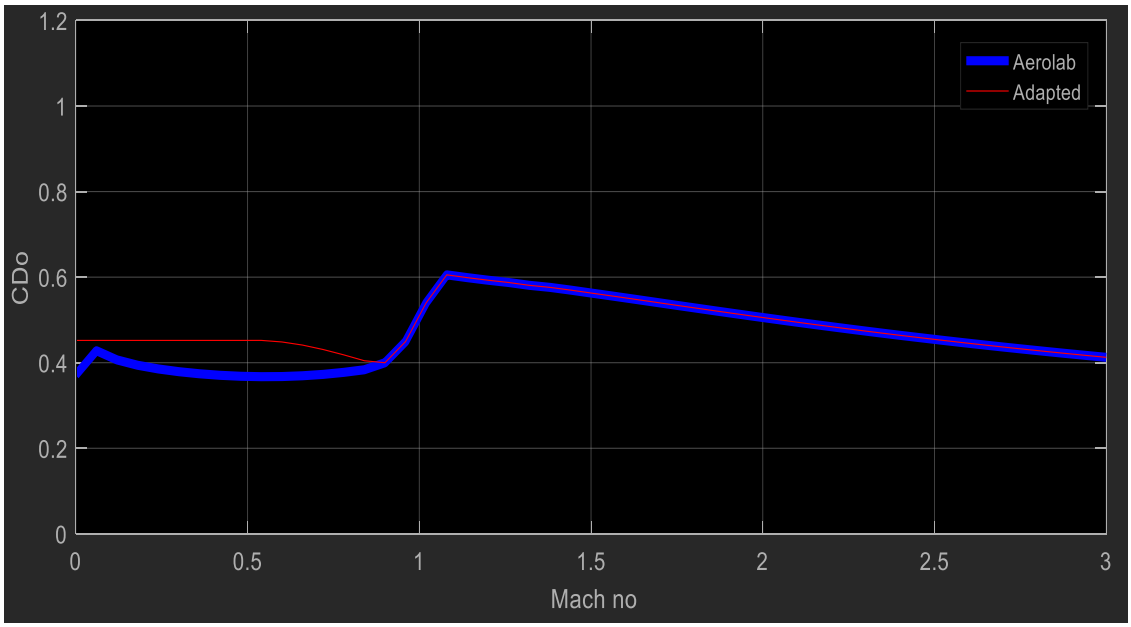


Figure 47. Estimated and tuned Coasting C_{Do} curves of FZ90 Rocket [A Landman, 2014]

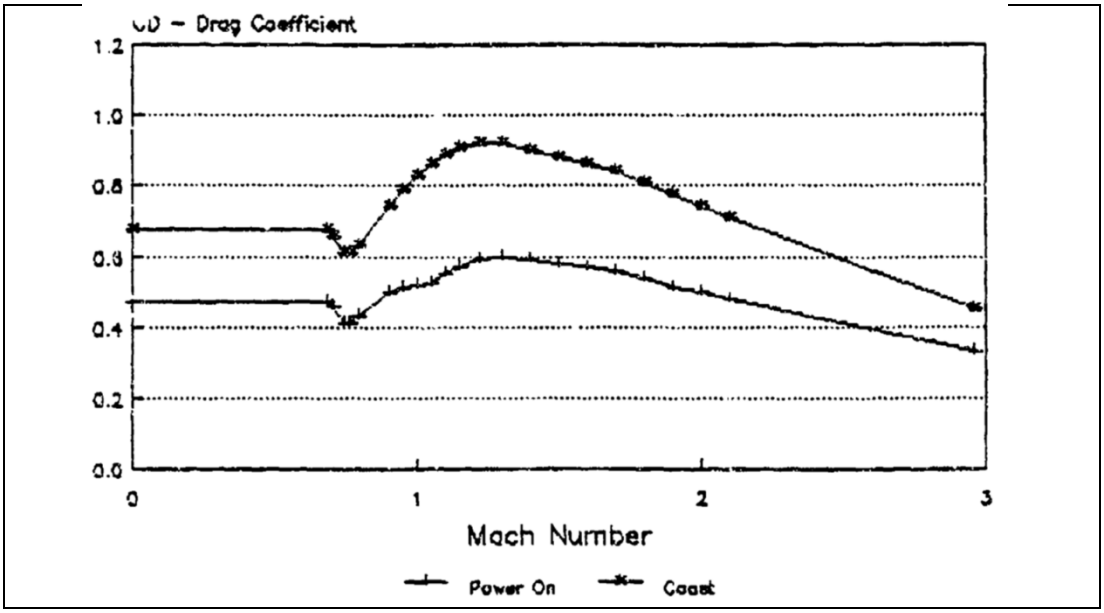


Figure 48. C_{Do} curves of MK-66 Rocket Motor & M261 Warhead [Dahlke and Batiuk, 1990]

To verify the validity of the blue curve of Figure 47 it was compared with the C_{Do} curves of the MK-66 Rocket Motor fitted with M261 High Explosive Warhead as described by *Dahlke* and *Batiuk* [21] and illustrated in Figure 48. The similarity between the two curves is evident albeit with differences such as that the C_{Do} curve of the MK-66/M261 Rocket is reduced by basebleed. This causes a reduction of about 0,18 in the overall C_{Do} curve of the MK-66/M261 Rocket during the **Boost** phase (rocket motor burning) compared to the **Coast** phase. In addition, the Coast C_{Do} curve of the MK-66/M261 Rocket is substantially higher, reaching a maximum value of 1,0 versus 0,6 in case of the blue curve of Figure 47. This is probably due to a difference in reference areas used in the two cases as shown later.

Note: *The difference between the drag curves of the MK66/M151 and MK66/M261 combinations is probably due to the transonic area rule. By employing this rule the drag of the MK-66/M151 combination can be reduced by widening the Fuze and M151 Warhead with a "clip-on fairing" as described by Hawley [12].*

According to *Dahlke* and *Batiuk* [21] the Coast curve of Figure 48 was derived from flight test data acquired from MK-66/M151 Rockets in the Coasting Phase of flight. This was probably done by adjusting the C_{Do} curve of a 6 DOF rocket model to match the observed data. The Power On curve of Figure 48 was probably estimated by adjusting the C_{Do} curve of a 6 DOF rocket model to match the maximum speed of the MK-66/M151 Rockets as observed during the Launch Phase.

In the case of the FZ90 Rocket a similar approach was followed by using a Tracking Radar to measure the trajectories of a number of FZ90/FZ181 Rockets launched from a Ground Jig at Overberg Test Range RSA. Because these tests were conducted from a Ground Jig, the effects of rotor downwash were eliminated. Some of these FZ90 Rockets were launched at super-elevation angles in the range 30° to 45° so that their trajectories were large (about 50 seconds flight time) and hence, allowed ample observations for making suitable adjustments to the C_{Do} curve of the FZ90 Rocket.

Figure 49. FZ90 Motor with FZ181 Warhead being launched from Ground Jig [Onno Sakkers, 1997]

When a FZ90 Rocket launched at Super Elevation of +30° was simulated with the **Rocket Launching System Model** of Figure 20 using the blue C_{Do} curve of Figure 47, the simulated trajectory of was found to overshoot the measured equivalent. In this simulation, the reference

surface area (A_o) of the Boggom Rocket was assumed to be 0,003848 m^2 as calculated by the AEROLAB program. This is simply the frontal area of a rocket that is 70mm in diameter in accordance with the FZ90 Rocket definition of paragraph 1.1.2.

Note: *The flight test data of the FZ90 Rocket as used here is not available in the open literature. The measurements were made by ANSYS Ltd RSA and was acquired by the Author during the course of his work at Denel Aeronautics*.

The overshoot result suggests that the reference surface area A_o as calculated by the AEROLAB program is too small. This is to be expected, since the stabilizer fins of the FZ90 Rocket must also represent some or other frontal surface area. However, they are not taken into account by the AEROLAB program. Hence, a new effective reference surface area was defined for the FZ90 Rocket:

$$A_{eff} = K_e A_o \qquad \textbf{Equation 23}$$

with A_{eff} the effective reference surface area of the FZ90 Rocket, K_{fe} a scaling factor and A_o the frontal cross-sectional area of the FZ90 Rocket body (0,003848 m^2) as calculated by the AEROLAB program. Recalculating the trajectory of the FZ90 Rocket assuming the flight test conditions and the blue curve of Figure 47 with K_e equal to 1.0, 1.2 and 1.4 results in the set of trajectories as illustrated in Figure 50. From this figure it followed that the value for K_e must be in the region of 1,35.

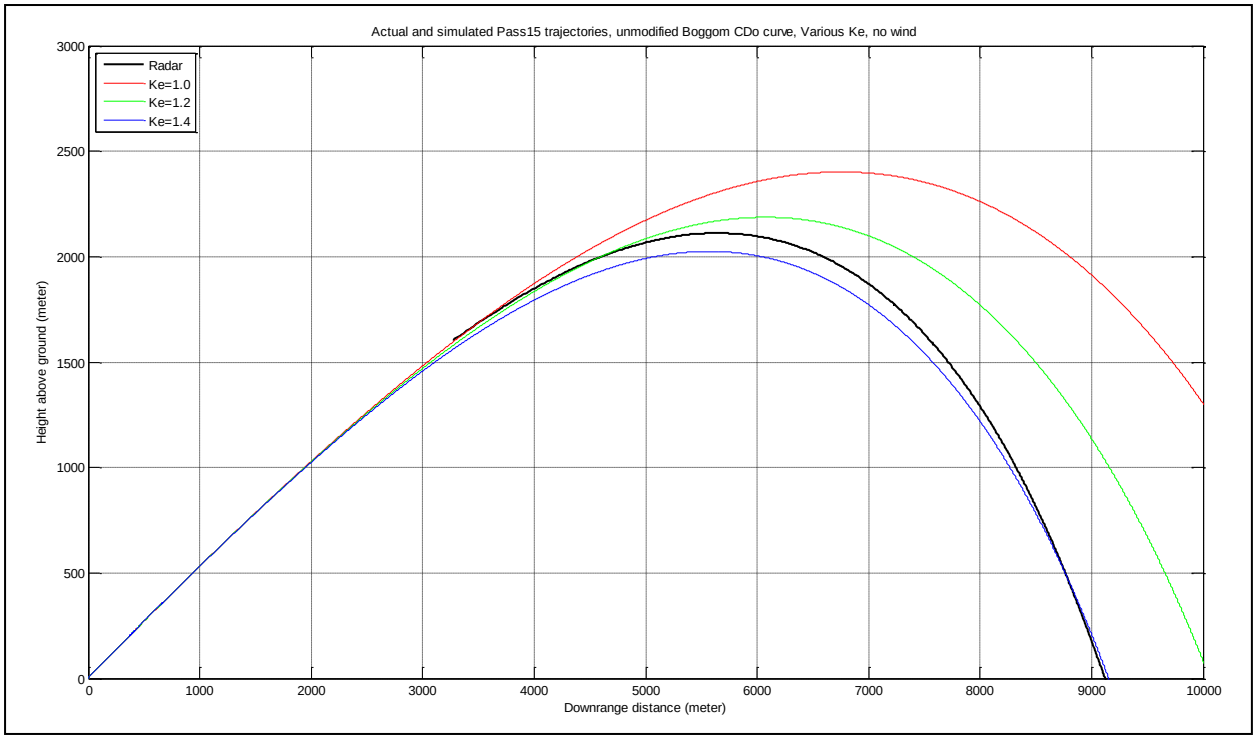


Figure 50. Actual and simulated Pass15 vertical trajectories, unmodified C_{Do}, SE = 30.0°, various K_e [A Landman, 2014]

Close observation of Figure 50 also indicated that the Rocket was launched at a Super Elevation slightly more than 30° (such a set-up error was very possible since the Ground Jig was not designed for adjusting the Super Elevation angle within accuracy of 0,1 degree). Some experimentation with the **Rocket Launching System Model** showed that the actual Super Elevation angle must have been close to 30,6°. Recalculating the trajectory of the FZ90 Rocket with Super Elevation angle of 30,6° and using the blue CD_o curve of Figure 47 with K_e equal to 1,37 resulted in the trajectory as illustrated in Figure 51. This choice of parameters gave a very

good fit from launch to apogee after which the drag force becomes too small. It is evident that at an elevation angle of 30° a change of 0,6° in elevation causes a change of about 100 m in apogee. Note that the range of the FZ90 Rocket is maximum at an elevation angle of about 30°. At smaller elevation angles the change in apogee is less.

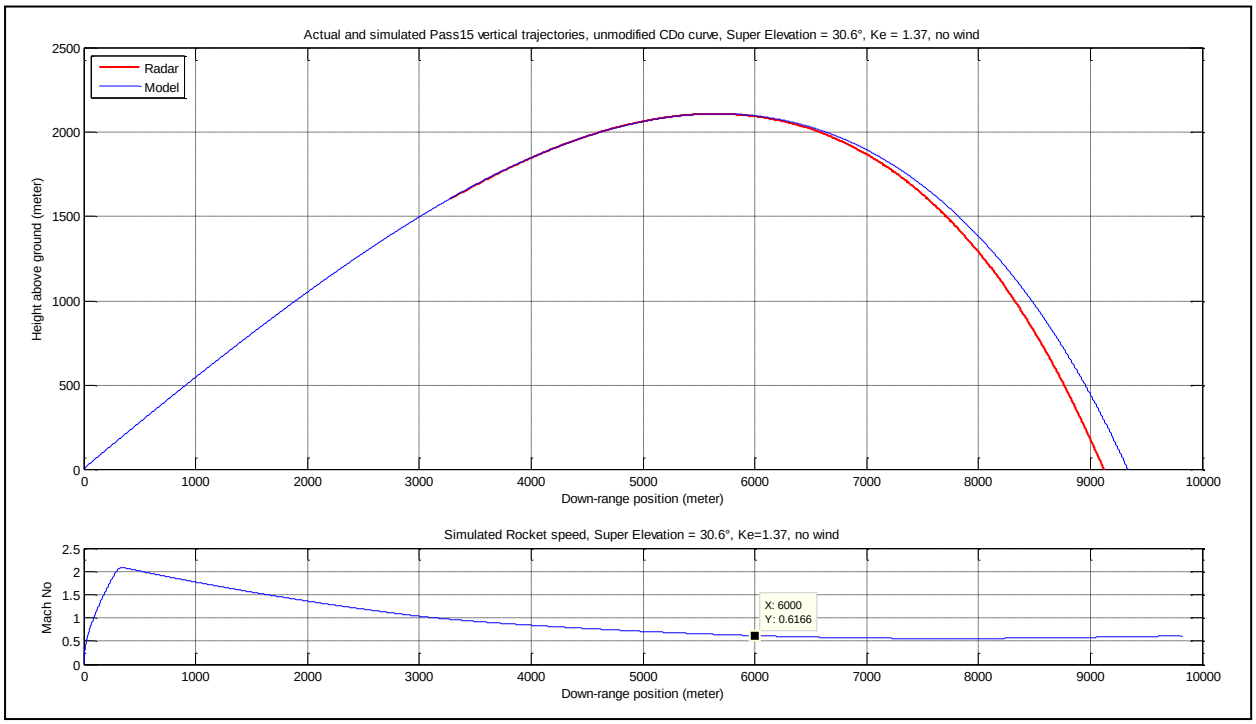


Figure 51. Actual and simulated Pass15 vertical trajectories, unmodified CDo, SE = 30.6°, K_e = 1.37 [A Landman, 2014]

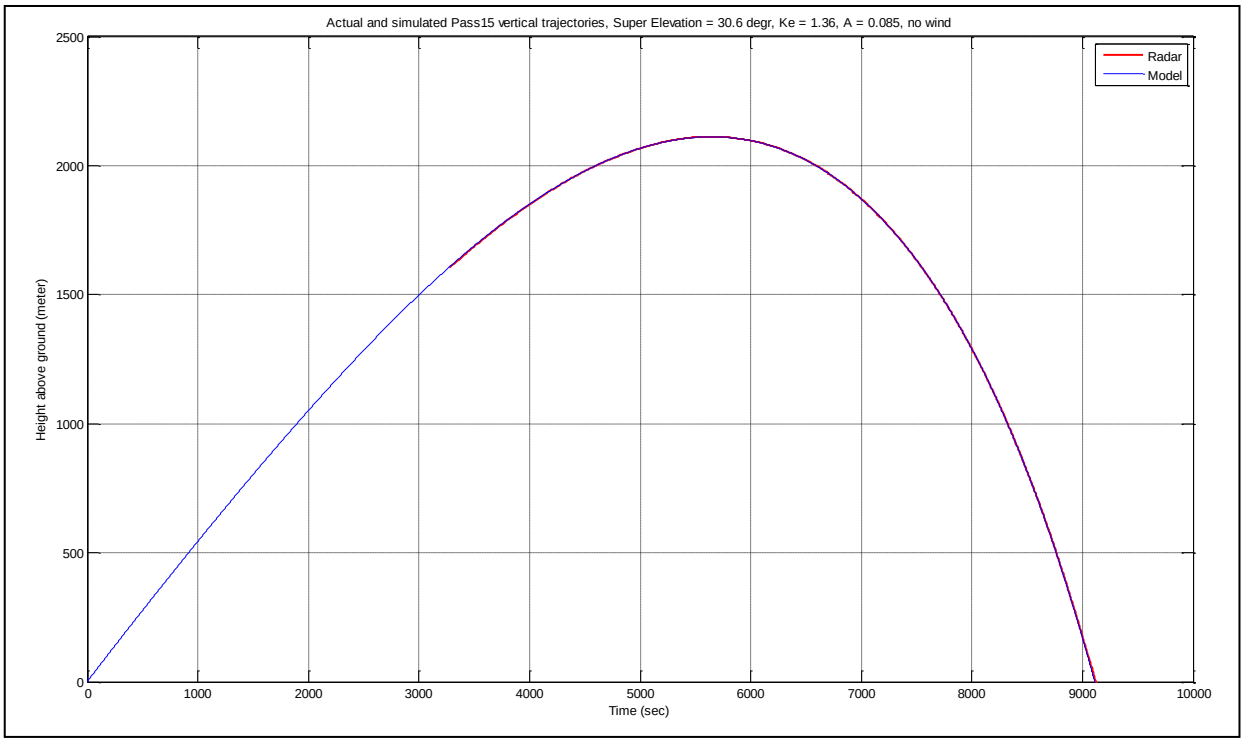


Figure 52. Actual and simulated Pass15 vertical trajectories, modified CDo with A = 0.085, SE = 30.6°, Ke = 1.37 [A Landman, 2014]

Inspection of Figure 51 showed that the AEROLAB program underestimated CD_o of the FZ90 Rocket at subsonic speeds, especially around Mach 0,5. This was to be expected since the AEROLAB program does not make provision for curved fins as used by the FZ90 Motor, forcing the FZ90 Rocket to be approximated with straight fins. In addition, there are also slight differences in shape between the nose cone of the FZ90 Rocket and those selectable in the AEROLAB program. Experimentation with the **Rocket Launching System Model** showed that adaptation of the original C_{Do} curve into the red curve of Figure 47 results in a very good fit between simulation and flight test data.

Recalculating the trajectory of the FZ90 Rocket with Super Elevation angle of 30,6° and using the red CD_o curve of Figure 47 with K_e equal to 1,37 results in the trajectory as illustrated in Figure 52. The Trajectory Height Error was calculated and found to be less than 2 meter over the entire trajectory.

It is evident that the use of a single Coasting C_{Do} curve does not have a major influence on the trajectory of the FZ90 Rocket. The reason for this can be found in Figure 51 – the launch stage is but a small fraction of the overall trajectory. Most of the trajectory is determined by the Coasting C_{Do} curve.

2.5.8.2 FZ90 Rocket Body Lift

To model the lift generated by the FZ90 Rocket body this study follows the assertion by *Anderson* [45] that all aerodynamic effects such as lift and drag forces acting on a body are caused by air (Space) generating a pressure and shear distribution over the surface of the body which, when summed, yields torque, drag and lift acting on the body. For the purpose of the **Rocket Launching System Model** as developed for this Chapter the lift force generated by the FZ90 Rocket Body is modelled with the DFBD as illustrated in Figure 53.

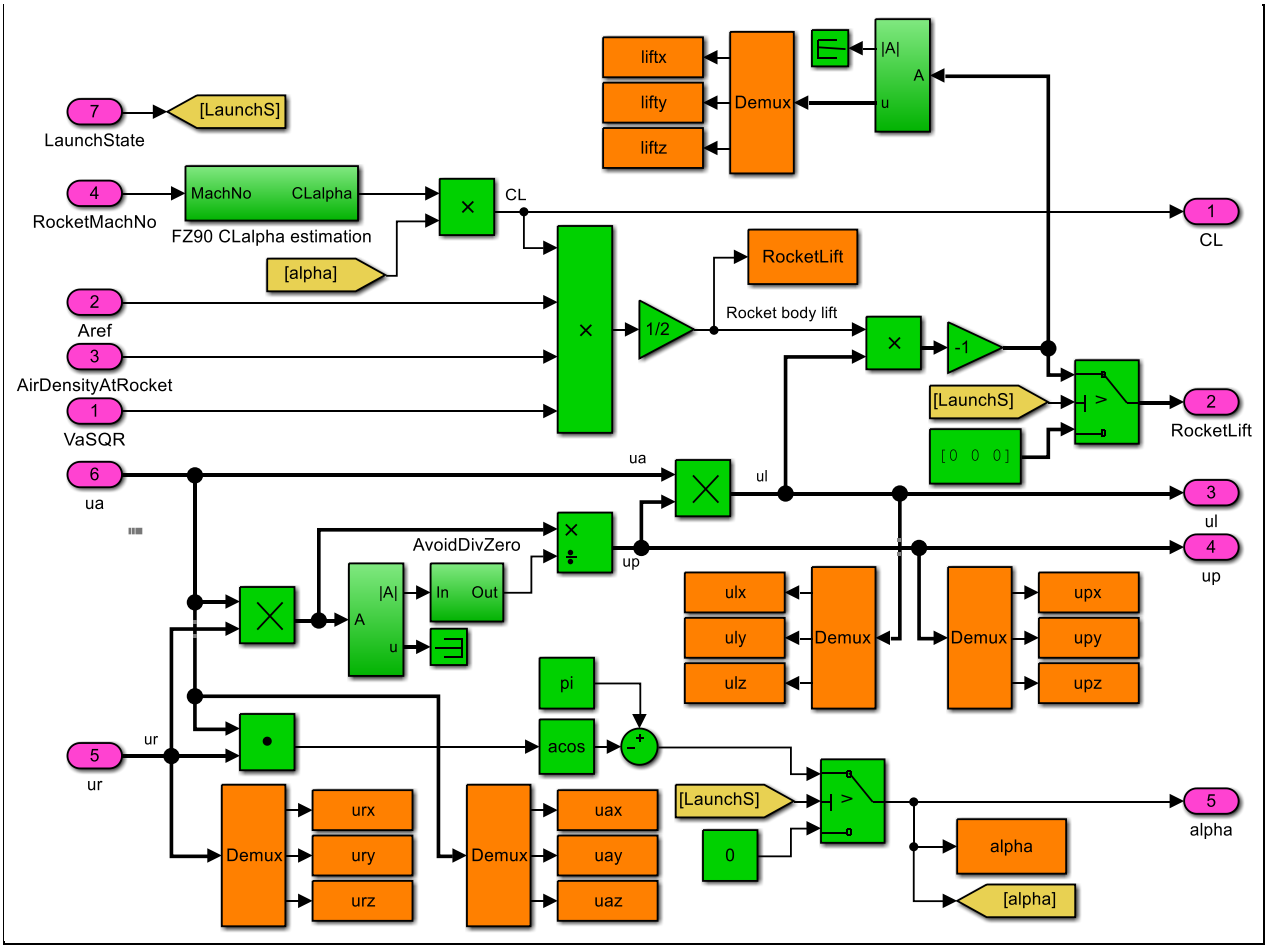


Figure 53. DFBD of FZ90 Body Lift Estimation block [A Landman, 2014]

As described by *Anderson* [45], the lift produced by a body moving through air (Space) is directly proportional to the square of the velocity with which the body moves through the air:

$$\mathbf{F}_L = \frac{\rho_y V_a^2 S_{ref} C_L}{2} \hat{e} \qquad \textbf{Equation 24}$$

with $\mathbf{F}_L$ the lift force in Newton acting on the body through its CoP, V_a the airspeed in m/s of the body, ρ_y the density of air in kg/m^3 in the vicinity of the body, S_{ref} some or other reference area in m^2 that represents the size of the body, C_L the lift coefficient of the body and $\hat{\boldsymbol{e}}$ a unit vector that designates the direction of the lift force. For the purposes of this study the forces acting on the FZ90 Rocket during free flght as implied by Equation 24 are illustrated in Figure 54.

Figure 54 illustrates the fact that in contrast to the drag force F_D, the lift force F_L is orthogonal to the direction at which a body moves through the air. In addition, the direction of $\boldsymbol{F}_L$ depends on the angle of attack α as illustrated in Figure 54. For such a configuration, unit vector $\boldsymbol{ue}$ can be calculated with two successive cross-products:

$$\mathbf{ue} = \mathbf{ua} \times (\mathbf{ua} \times \mathbf{ur}) \qquad \textbf{Equation 25}$$

with $\boldsymbol{ur}$ a unit vector along the centre line of the FZ90 Rocket, pointing from the tail to the nose. The density ρ_r of the air in the immediate vicinity of the rocket is calculated as described in paragraph 2.5.8.5.

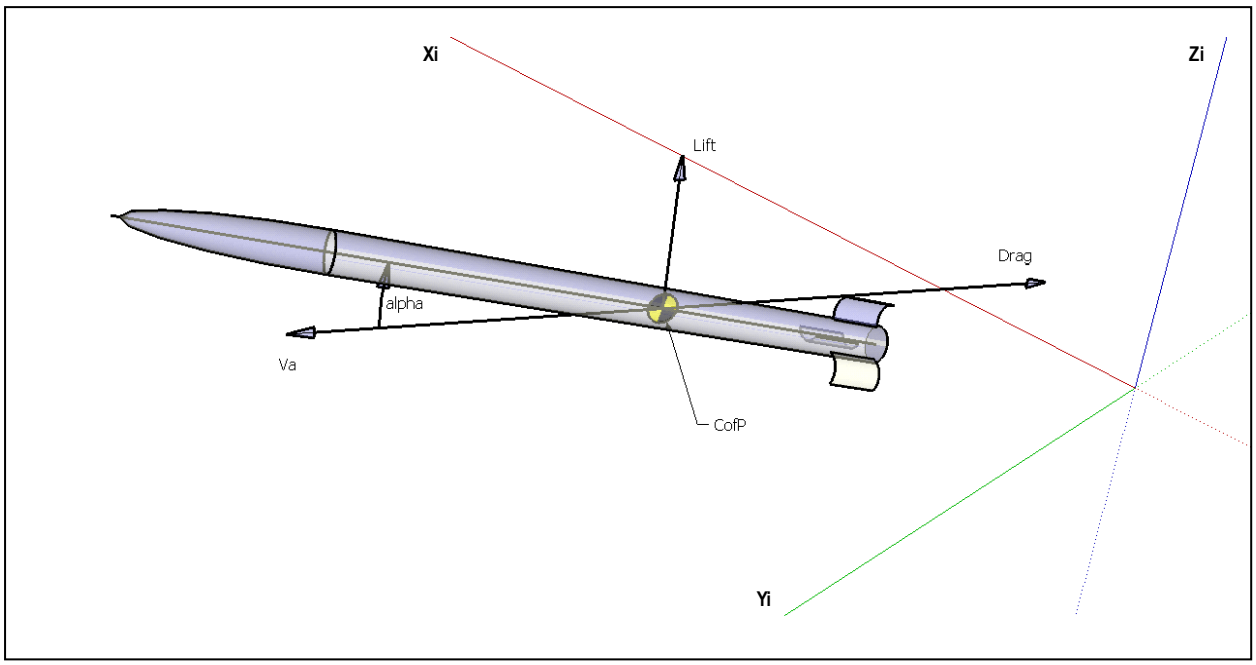


Figure 54. Drag and Lift forces acting on FZ90 Rocket [A Landman, 2014]

As described by *Anderson* [45] the lift coefficient C_L is dependant on Mach number and angle of attack. At a given airspeed C_L varies linearly with α provided the angle of attack does not exceed the stall angle (usually about 16°). The same relationship is also applicable to a lifting body such as the FZ90 Rocket, so that we have:

$$C_L = C_{L\alpha}\alpha + C_{Lo} \qquad \textbf{Equation 26}$$

with $C_{L\alpha}$ the *Lift Curve Slope Coefficient.* Note that $C_{L\alpha}$ has a value of 2π based on Thin Airfoil Theory as developed by *Munk* [47]. C_{Lo} is the lift coefficient at zero angle of attack, and is zero for a symmetric body such as the FZ90 Rocket. In reality $C_{L\alpha}$ changes with Mach number and was estimated for the FZ90 Rocket using the AEROLAB program which produced the curve of Figure 55 (calculated with respect to the frontal area of the rocket). The **FZ90 CLalpha Estimator** block as used in the DFBD of in Figure 53 contains this curve, implemented as a lookup table and coded as an S-function for precompilation and use in Simulink.

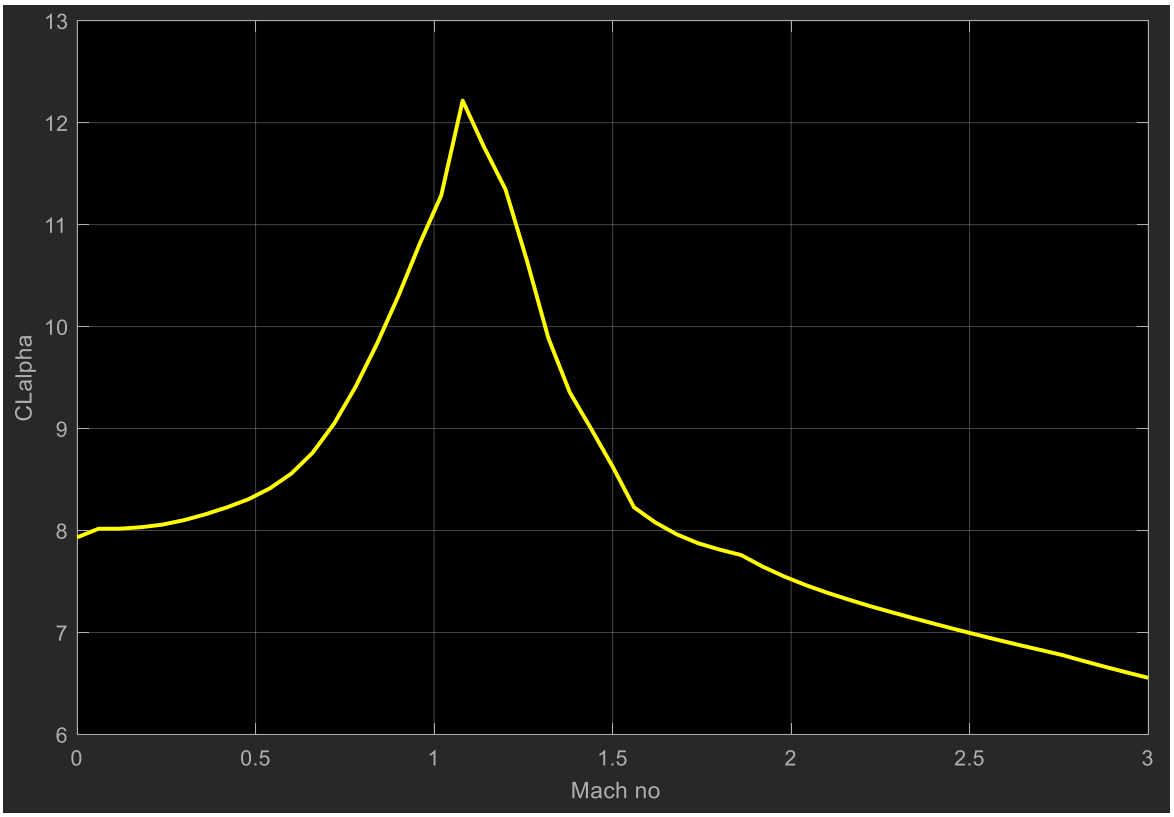


Figure 55. Lift curve slope coefficient of FZ90 Rocket [A Landman, 2014]

The angle of attack of air flow over the FZ90 Rocket is calculated by making use of the scalar product between the rocket unit vector $\boldsymbol{ua}$ and airspeed unit vector $\boldsymbol{ur}$:

$$\begin{aligned} & \cos(\alpha) = \mathbf{ua} \cdot \mathbf{ur} \\ \Rightarrow\ & \cos(\alpha) = uax \cdot urx + uay \cdot ury + uaz \cdot urz \\ \Rightarrow\ & \alpha = \operatorname{acos}(uax \cdot urx + uay \cdot ury + uaz \cdot urz) \end{aligned}$$ **Equation 27**

with *uax*, *uay* and *uaz* the components of unit vector $\boldsymbol{ua}$ in Inertial Axes. Similarly *urx*, *ury* and *urz* represent the components of unit vector $\boldsymbol{ur}$ in Inertial Axes.

Note: *The reference area S_{ref} used in Equation 24 is taken as the same effective cross-sectional area used in Equation 23 because that is the assumption made by the AEROLAB program.*

2.5.8.3 FZ90 Rocket Body Torque

In order to model the dynamic behaviour of the FZ90 Rocket, the **FZ90 Rocket Body Torque Estimation** block estimates three rolling moments generated by different parts of the body of the FZ90 Rocket.

2.5.8.3.1 Roll Moment

The first rolling torque generated by the FZ90 Rocket Body originates from the Wrap-Around Fins as illustrated in Figure 56 and Figure 57. The torque generated by these fins are in

opposition to the torque generated by the FZ90 Rocket Motor and are scaled such as to result in roll reversal during flight as illustrated in Figure 17 for a MK-66 MOD1 Rocket.

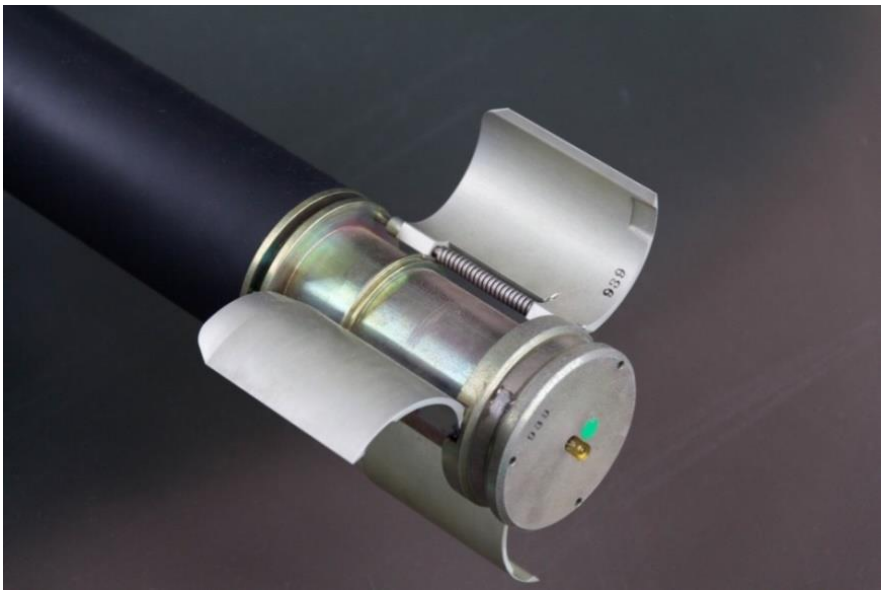


Figure 56. Wind tunnel model: MK-66 MOD1 showing bevels and tabs [Author unknown]

Figure 57. Wrap-Around Fin and Nozzle of a recovered FZ90 Rocket Motor [A Landman, 2017]

Mermagen and *Oskay* [23] describe this arrangement as a compromise solution in the case of the MK-66 MOD1 Rocket to ensure rapid spin-up (reaching about 7 Hz upon leaving the Launch Tube) to counter the effects of downwash, tip-off and thrust misalignment, while limiting maximum spin rate to remain compatable with the arming mechanisms of the M423 Fuze and M261 Warhead. In addition, roll rates of 8 Hz in supersonic flight and 4 Hz in subsonic flight must be avoided to prevent pitch/roll lock-in. These limits are illustrated in Figure 61.

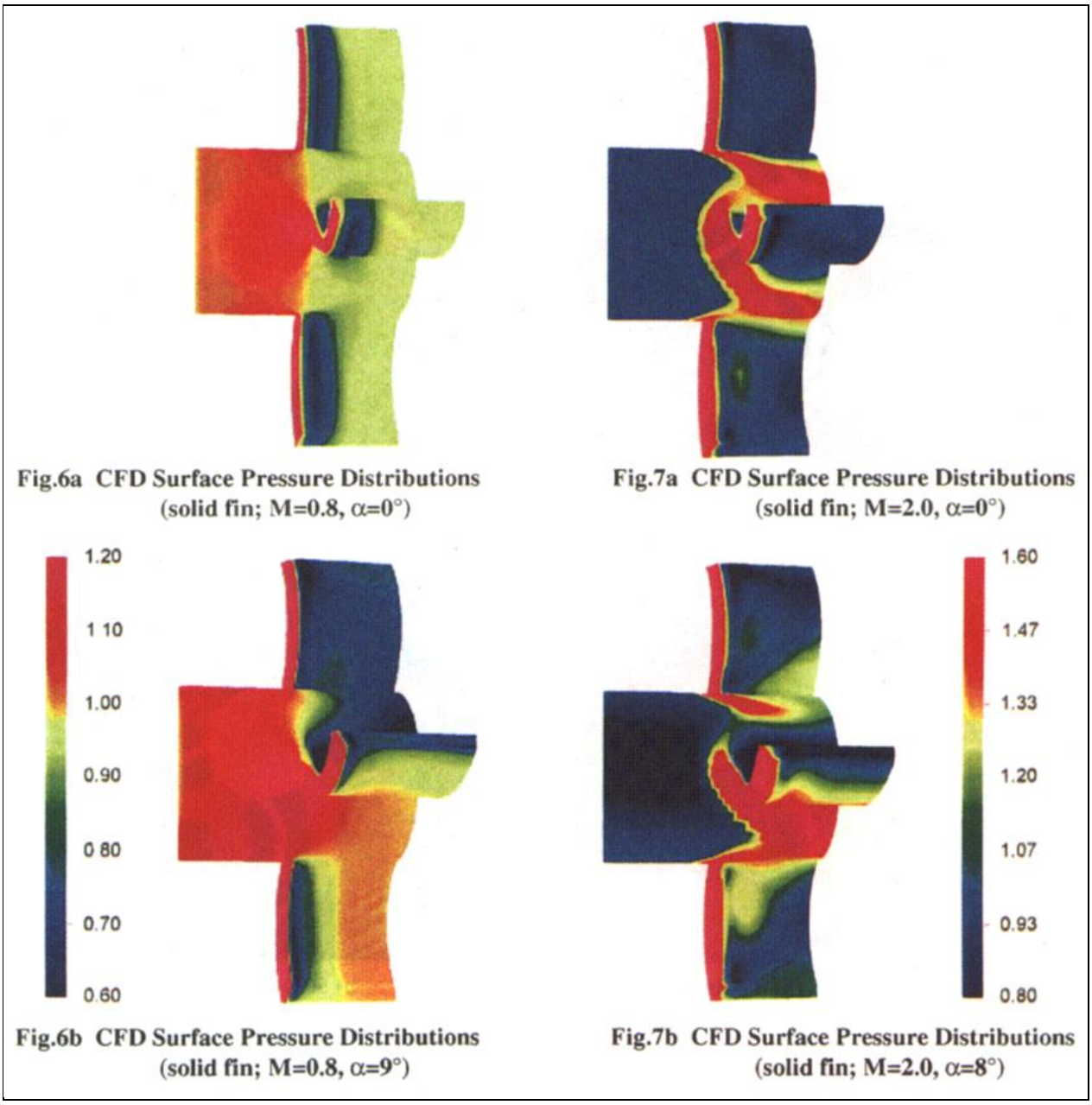


Fig.6a CFD Surface Pressure Distributions (solid fin; M=0.8, α=0°)

Fig.7a CFD Surface Pressure Distributions (solid fin; M=2.0, α=0°)

Fig.6b CFD Surface Pressure Distributions (solid fin; M=0.8, α=9°)

Fig.7b CFD Surface Pressure Distributions (solid fin; M=2.0, α=8°)

Figure 58. Pressure distribution over rocket fitted with Wrap Around Fin [Berner, Abate and Dupuis, 1998]

The open literature contains a wide range of papers on the characteristics of Wrap-Around Fins. For the purpose of this study only those applicable in characterizing the FZ90 Rocket are described. *MIL-HDBK-762(MI)* [29], *Dahlke* [48] and *Mandić* [49] describe rolling moment at zero incidence as an enigmatic characteristic of Wrap-Around Fins. This effect causes a Wrap-Around Fin to generate lift at zero incidence even though it may not be fitted with any bevel, tab or cut-out. This lift is usually directed toward the centre of curvature of the wing during subsonic flight and vise-versa during supersonic flight. Bevels, tabs, cut-outs and body step-downs affect the lift and were tested in various combinations during development of the Wrap-Around Fins of the MK-66 Rocket Motor, yielding the MOD1 configuration as illustrated in Figure 56.

Other modifications such as screening of the detonator circuit to comply with Hazards of Electromagnetic Radiation to Ordnance (HERO) requirements and addition of a K_2SO_4 rod to avoid aircraft engine flame-out resulted in a series of Hydra-70 rocket motors with different characteristics (MOD4 being the version intended for helicopter use). The MOD1, MOD2,

MOD3, MOD4, MOD5 and MOD6 rockets all use the same fin as illustrated in Figure 56. The FZ90 Rocket Motor seems to be a close copy of the MK-66 MOD1 Rocket Motor as illustrated in Figure 57.

Wilks [50] used Evvard's theory to explain the rolling moment at zero incidence characteristics of rockets fitted with Wrap-Around Fins. His analysis involves different zones of influence created by intersections between the concave and convex surfaces of the fins and Mach cones originating from the leading edges of the fins. *Berner*, *Abate* and *Dupuis* [22] explained the effect by solving the flow field of a rocket (without step-down), fitted with Wrap-Around Fins using a Computational Fluid Dynamics model, indicating an uneven pressure distribution over the fins that generates a longitudinal torque at zero Angle of Attack. Their results are illustrated in Figure 58, showing that for subsonic flight at zero Angle of Attack a larger region of low pressure exists on the concave side versus the convex side of the wings. For supersonic flight the pressure differences switch around. At an Angle of Attack of about 8° the effect is even more pronounced. It is clear from Figure 58 that at supersonic speeds the rolling moment created by such a rocket is the joint response of rocket body and all the fins – they cannot be analyzed as separate entities.

Dahlke [48], Mandić [49] as well as *Berner*, *Abate* and *Dupuis* [22] all used wind tunnels to measure the effects of various features including bevels, tabs cut-outs, wing thickness, aspect ratio, sweep angle and step-down body configurations (as in Figure 56 and Figure 57) for the purpose of developing a model suitable to predict the behaviour of rockets fitted with Wrap-Around Fins. The rolling moment coefficient of a Wrap-Around Fin design as measured by *Dahlke* [48] and designated as *B1F16* (sweeped back fin with bevelled 45° leading edge) at zero Angle of Attack is illustrated in Figure 59 and illustrates the rapid change in rolling moment that occurs in the vicinity of Mach one.

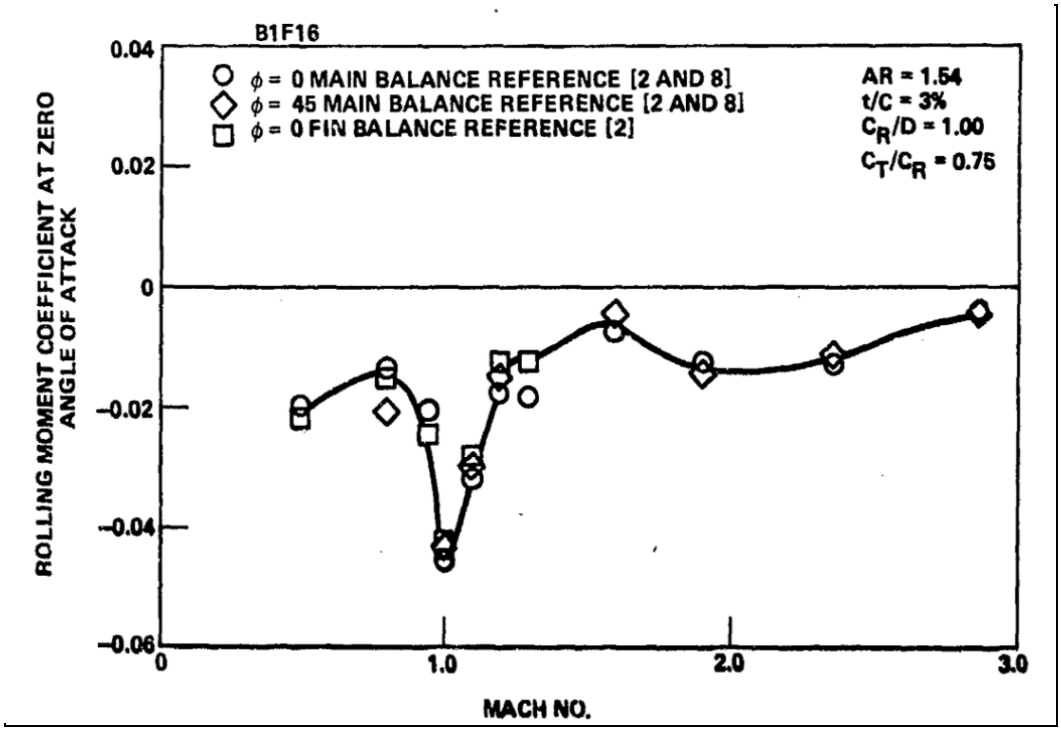


Figure 59. Roll Moment Coefficient versus Mach no for B1F16 wing [Dahlke, 1976]

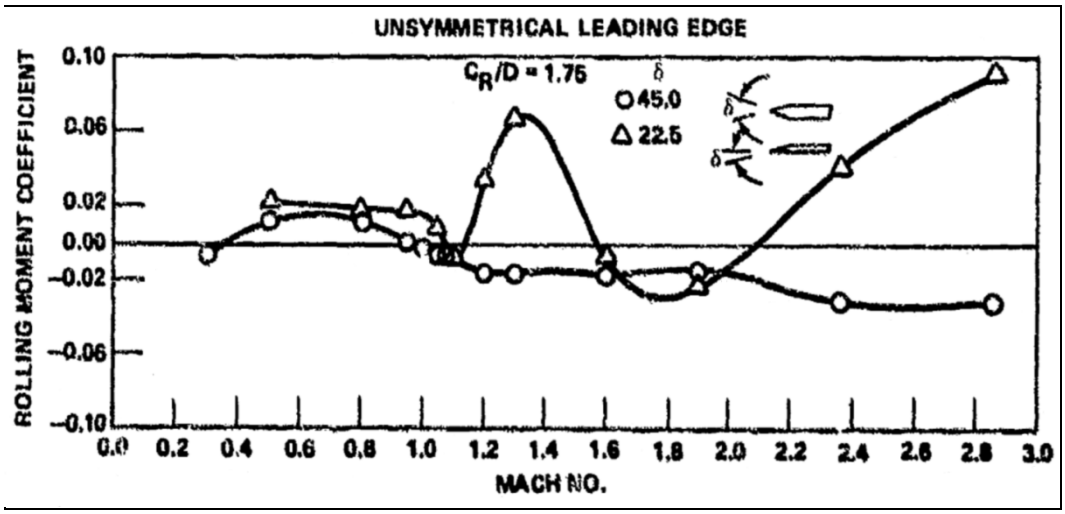


Figure 60. Influence of beveling on Roll Moment Coefficient [Dahlke, 1976]

The effect that bevelling the leading edge of the convex side of a rectangular fin such as those as illustrated in Figure 56 has on the Rolling Moment Coefficient was measured by *Dahlke* [48] with result as illustrated in Figure 60. It is clear that small changes to the Wrap-Around Fins have big effects on the Roll Moment Coefficient of the MK-66 Rocket. *Dahlke* and *Batiuk* [21] describe how this characteristic of the Wrap-Around Fin was utilized to solve one of the problems encountered with the MK-66 MOD0 Rocket, being a reduction in range as a result of coning (which increases drag) that developed during flight. This problem is illustrated in Figure 61 which shows the spin limits of the M423 Impact Fuze and the M261 Warhead as well as the spins that should be avoided to prevent pitch/roll lock-in. It is clear that the MK-66 MOD0 Rocket enters the pitch/roll lock-in zone after 6 seconds which is then bound to lead to coning (due to dynamic unbalance) with concomitant reduction in range.

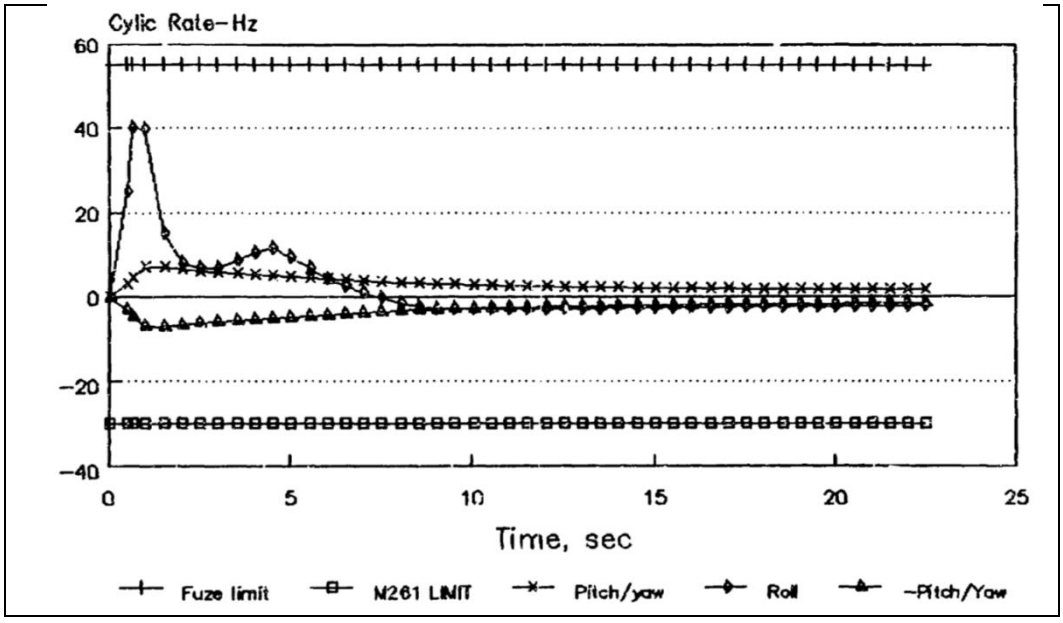


Figure 61. Typical evolution of spin and velocity of MK-66 MOD0 Rocket [Dahlke and Batiuk, 1990]

Dahlke and *Batiuk* [21] describe how the lock-in problem of the MK-66 MOD0 Rocket Motor was solved by introducing a bevel on the convex side of the leading edge and a tab on the concave side of trailing edge of the fins as illustrated in Figure 56. The effects of these modifications (as determined in wind tunnel with stationary rocket), are illustrated in Figure 62 - the bevel on the

convex side of the leading edge is only effective at supersonic speeds while the tab on the concave side of the trailing edge is effective at subsonic speeds. Combining the Roll Moments of the Wrap Around Fin, bevel and tab yields the overall Roll Moment Coefficient of the MK-66 MOD1 Rocket Motor as illustrated at bottom of Figure 62.

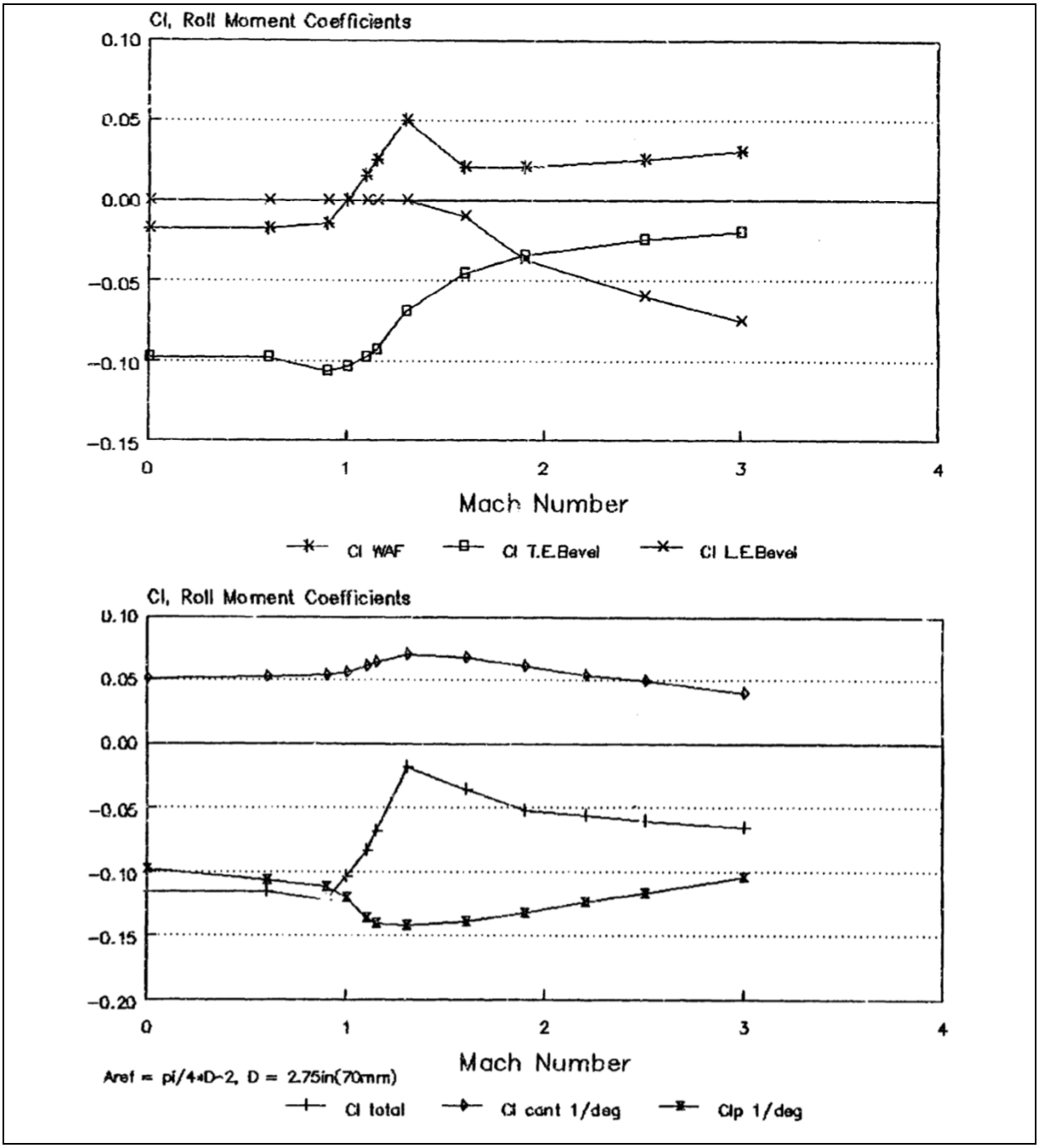
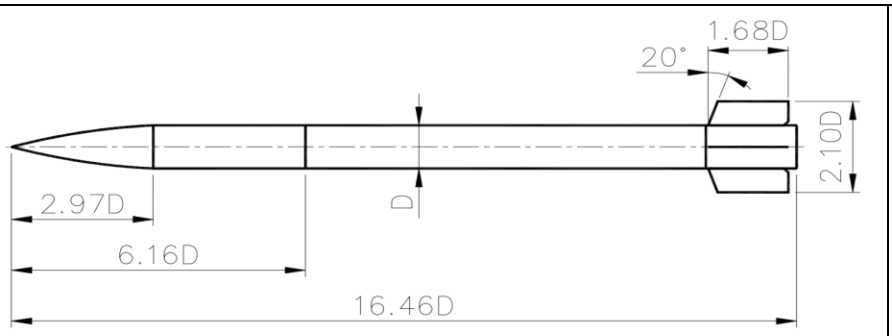
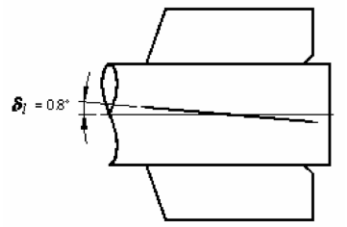
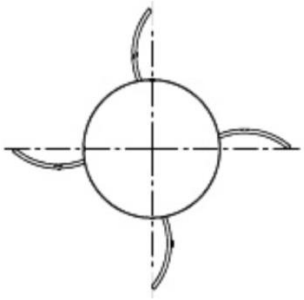


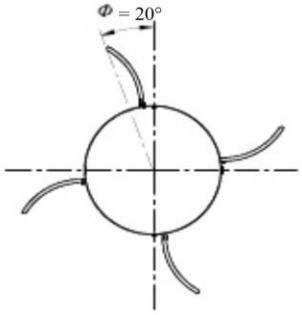
Figure 62. MK-66 Roll Moment Coefficients [Dahlke and Batiuk, 1990]

The modifications as described above yield the typical evolution of spin of the MK-66 MOD1 Rocket with M151 Warhead as illustrated in Figure 17. This configuration produce just the right amount of torque as function of Mach number to cause rapid roll reversal through the pitch/roll lock-in zone without exceeding the spin limits of the M423 Impact Fuze and M261 Warhead. Note that Figure 17 illustrates the dynamic behaviour of the MK-66 MOD1 Rocket Motor with M151 Warhead while Figure 62 illustrates the static behaviour of the same rocket.

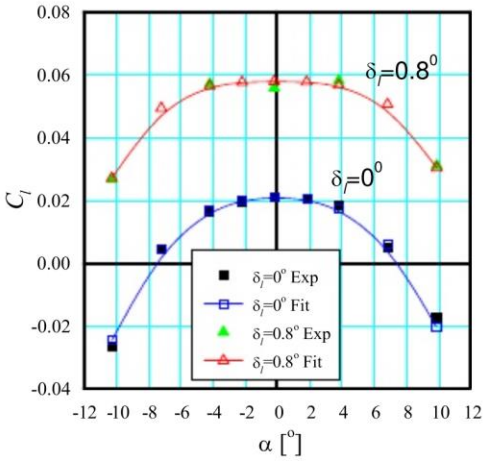




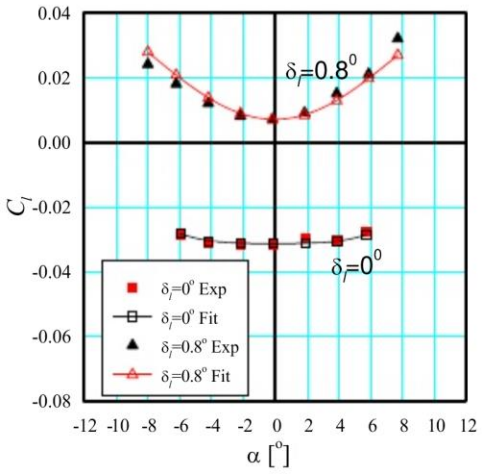




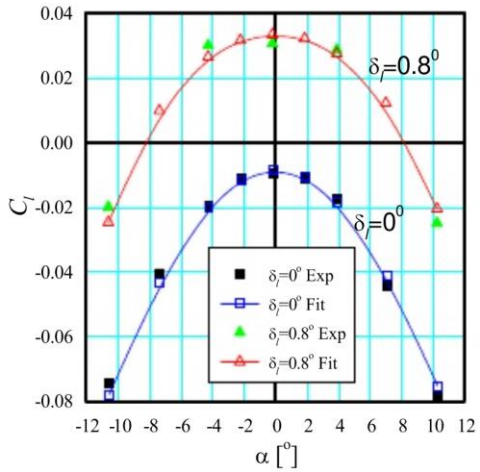
MACH 0,8



MACH 0,8



MACH 2,0



MACH 2,0

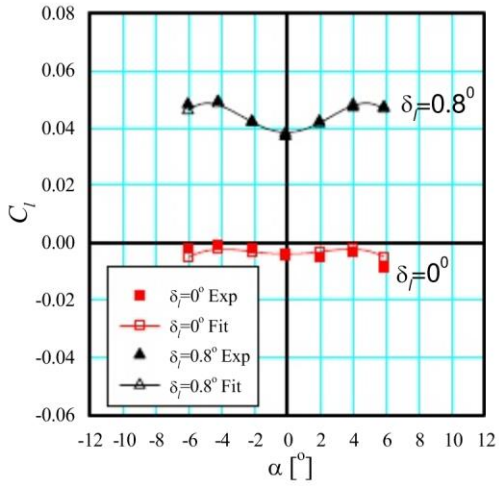


Figure 63. Roll Moment Coefficient of Wrap Around Fin versus pitch and cant [Mandić, 2006]

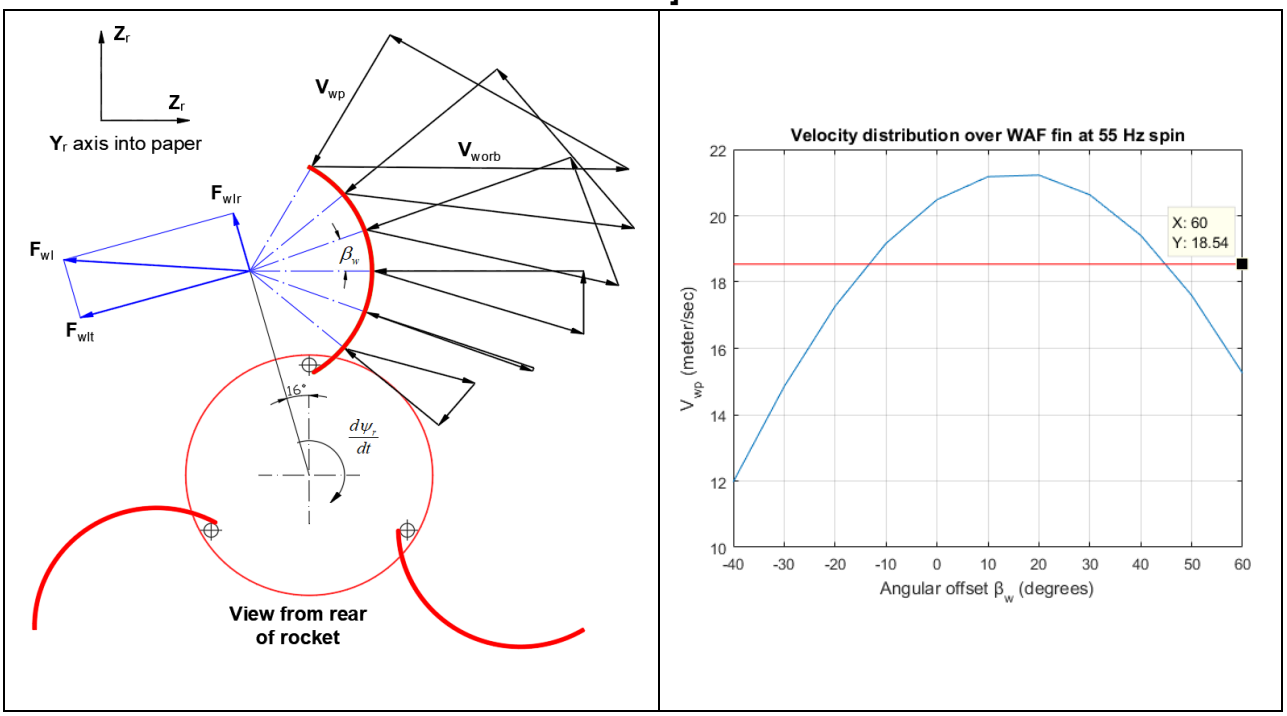


Figure 64. Velocity distribution over Wrap Around Fin due to rocket spin [A Landman, 2017]

It is evident that the rolling moment generated by the Wrap-Around Fins is dependant on both spin and forward motion of the FZ90 Rocket. To illustrate this effect the distribution of orbital velocities over the surface of a fin of the FZ90 Rocket spinning at 55 Hz is illustrated in Figure 64. The component $\boldsymbol{V}_{wp}$ of this velocity distribution normal to the fin surface generates lift acting through the "focus" of the fin as shown. Inspection of the figure shows $\boldsymbol{V}_{wp}$ has a maximum value of about 28 m/s at maximum allowable spin rate of 55 Hz. This implies an Angle of Attack of about 2,3° for this part of the fin with the FZ90 Rocket travelling at Mach two (maximum velocity – see Figure 61). For lower rocket velocities the Angle of Attack will be even larger. Such a large Angle of Attack is certain to affect the roll moment generated by the Wrap Around Fin.

Mandić [49] performed wind tunnel measurements on rocket models fitted with Wrap Around Fin canted as illustrated in Figure 63. The results of his study demonstrate at least two characteristics of Wrap-Around Fins – non-zero rolling moments at zero incidence and the effect of cant angle δ_l on rolling moment. The work of *Mandić* [49] indicates that the Rocket Model of Figure 63 has a $C_{l\delta}$ value of 0,05/degree. This agrees very well with the average value of $C_{l\delta}$ as function of Mach number for the MK-66 Rocket as reported by *Dahlke* and *Batiuk* [21] and illustrated in Figure 62.

Note: *The value of $C_{l\delta}$ is positive assuming δ_l is measured as shown in Figure 63 with the rocket facing into wind as illustrated in Figure 64.*

Note: *The assumption that $C_{l\delta}$ is constant for the MK-66 MOD1 Rocket is based on the work by Wilks [50] which shows that the lift coefficient of a Wrap Around Fin is directly proportional to Angle of Attack. It is also based on the fact that the lift coefficient curve of most aerofoils such as Clark Y are usually close to linear (constant slope) below stall speed.*

The Spin Roll Moment Estimation block calculates Roll Moment Coefficient C_{ms} caused by non-zero Angle of Attack as seen by the Wrap Around Fin. This moment is calculated by interpolating CL_{Total} and $C_{l\delta}$ as function of Mach number M_r based on the data in Table 5 of *Dahlke* and *Batiuk* [21]. The results are then used to calculate C_{ms} with a linear relationship using CL_{Total} as offset and cant angle δ_l as independent variable with $C_{l\delta}$ as constant of proportionality. The resulting 2D surface is illustrated in Figure 65.

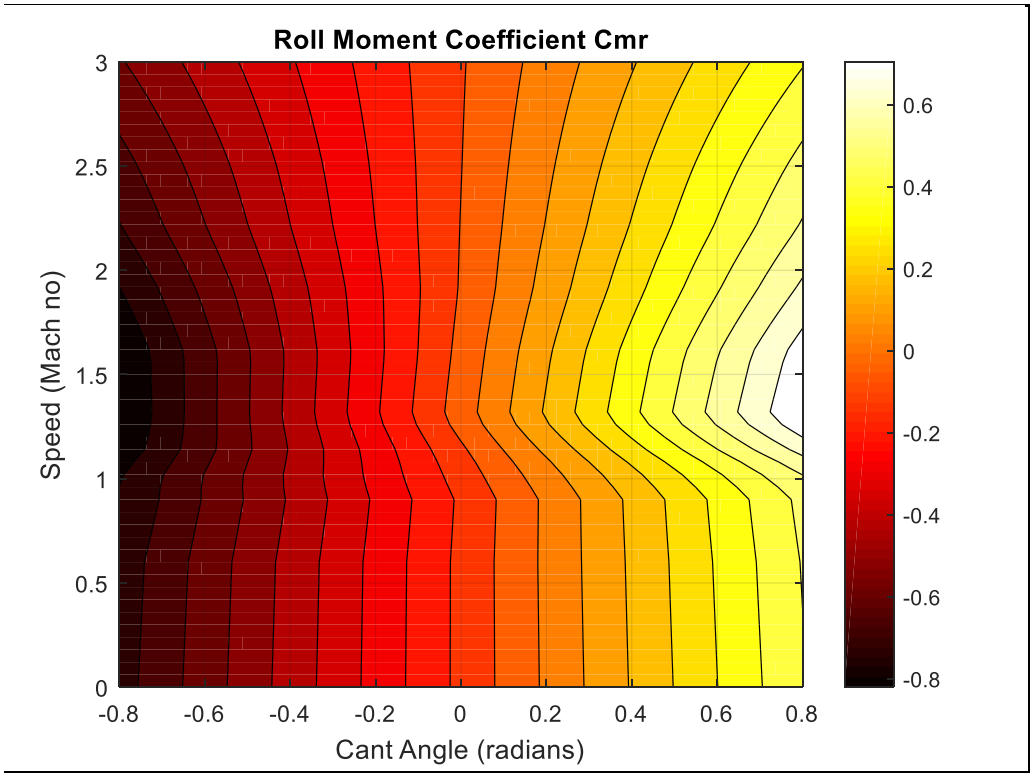


Figure 65. Roll Moment Coefficient, MK-66 MOD1 Rocket [Dahlke and Batiuk, 1990]

Mandić [49] as well as *Dahlke* and *Batiuk* [21] performed their measurements by canting the entire Wrap Around Fin as illustrated in Figure 63 while spin causes the Angle of Attack to vary over the Wrap Around Fin as illustrated in Figure 64. Hence, a given spin rate should translate to an effective cant angle – probably in a linear fashion based on the work by *Wilks* [50].

To determine the required relationships the relative air velocity $\boldsymbol{V}_a$ of the FZ90 Rocket in Inertial Axes is calculated as follows:

$$\mathbf{V}_a = \mathbf{V}_w - \mathbf{V}_r \qquad \textbf{Equation 28}$$

with $\boldsymbol{V}_r$ the velocity of the FZ90 Rocket in Inertial Axes and $\boldsymbol{V}_w$ the wind velocity in Inertial Axes. Hence, the relative air velocity $\boldsymbol{V}_{ar}$ of the FZ90 Rocket in Rocket Axes is given by:

$$\mathbf{V}_{ar} = IDC_{ZXY}(\phi_r, \theta_r, \psi_r) * \mathbf{V}_a \qquad \textbf{Equation 29}$$

with $IDC_{ZXY}(\phi_r, \theta_r, \psi_r)$ the inverse ZXY direction cosine matrix[5] resulting from yawing, pitching and rolling the FZ90 Rocket through the angles ϕ_r, θ_r and ψ_r in the same order. Inspection of Figure 64 shows the average wind speed normal to the Wrap Around Fin surface is 18,5 m/s with the FZ90 Rocket spinning at 55 Hz. This speed is directly proportional to the spin rate of

[5] See Appendix A

the FZ90 Rocket so that the "effective" wind speed normal to the Wrap Around Fin surface V_{wna} must be given by:

$$V_{wna} = k_{\delta}\left(\frac{d\psi_r}{dt}\right) \qquad \textbf{Equation 30}$$

with $\left(\frac{d\psi_r}{dt}\right)$ measured in rad/sec and k_{δ} an efficiency scale factor that must be determined from experimental data for a given rocket configuration.

Hence, the "effective" cant angle δ_l as seen by the Wrap Around Fin of the FZ90 Rocket is given by:

$$\delta_l = atan\left[\frac{k_{\delta}\left(\frac{d\psi_r}{dt}\right)}{V_{ary}}\right] \qquad \textbf{Equation 31}$$

with V_{ary} the component of $\boldsymbol{V}_{ar}$ along the **Y**r axis of Rocket Axes and δ_l the effective cant angle (in radians) as illustrated in Figure 63.

2.5.8.3.2 Pitch Moment

The second torque generated by the FZ90 Rocket Body is induced by perturbation of rocket centre-line with respect to direction of travel. Such a condition results in a non-zero Angle of Attack which leads to lifting force acting through the Centre of Pressure. Since for the FZ90 Rocket the Centre of Pressure is located about 100 mm behind the Centre of Mass, this results in a torque that opposes the perturbation, thus ensuring longitudinal stability of the FZ90 Rocket (but also making it very susceptible to wind and gusts).

Since the lift force acting on the FZ90 Rocket is already calculated as described in paragraph 2.5.8.2 the pitch moment (using Static Pitch Moment Coefficient C_{mp}) of the FZ90 Rocket is not calculated here per se.

2.5.8.3.3 Aerodynamic Damping Moment

The third torque generated by the FZ90 Rocket Body is caused by air resistance to rotation of the FZ90 Rocket in pitch. This torque is proportional to the rate of change in rocket pitch, making it dissipative in nature that ensures that perturbations in pitch decay. The aerodynamic damping moment for the FZ90 Rocket is given by the following:

$$M_{AD} = qAd\left(\frac{C_{Mq}D_r}{2V_{ary}}Q_r\right) \qquad \textbf{Equation 32}$$

with *qAd* a common dynamic pressure term as calculated in paragraph 2.5.8.3.5, C_{Mq} the Aerodynamic Damping Coefficient, D_r the reference diameter (0,07 meter) of the FZ90 Rocket, V_{ary} the component of $\boldsymbol{V}_{ar}$ along the **Y**r rocket axis and Q_r the rate of change in rocket pitch. The Aerodynamic Damping Coefficient C_{Mq} is a function of Mach number as described by *Dahlke* and *Batiuk* [21] and is illustrated in Figure 66 for the MK-66 Rocket Motor in combination with the M151 and M261 warheads. For the purpose of this study we assumed the

Aerodynamic Damping Coefficients of the FZ90 Rocket to be equivalent to that of the MK-66/M151 combination.

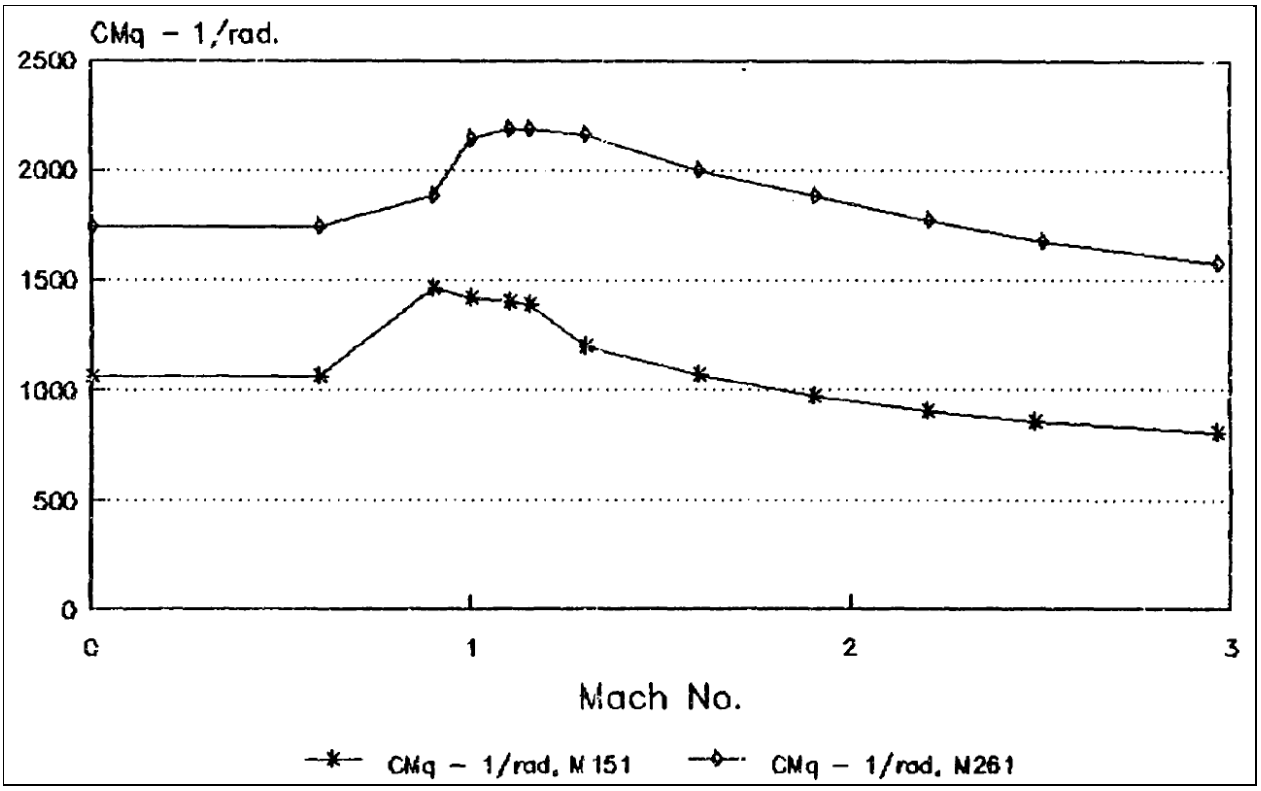


Figure 66. Aerodynamic Damping coefficient of MK-66 Rocket [Dahlke and Batiuk, 1990]

The direction of the Aerodynamic Damping Moment was determined by normalizing the cross-product between unit vector **ur** and unit vector **ua** to yield unit vector **up** as illustrated in Figure 53. Note that the Aerodynamic Damping Moment is calculated directly in Inertial Axes while the Roll Moment generated by Wrap Around Fin is calculated in Rocket Axes and has to be converted into Inertial Axes before use.

2.5.8.3.4 Side Moment

The fourth torque generated by the FZ90 Rocket Body is caused by a small side force generated by the Wrap Around Fin at non-zero angle of attack. The magnitude and effect of this side force was measured by *Berner*, *Abate* and *Dupuis* [22] for a very large wind-tunnel model with four Wrap Around Fin. Their observations show various cork-screw motions that bear resemblance to that as illustrated in Figure 15 for the FZ90 Rocket.

No Side Moment Coefficient data for the MK-66 Rocket or FZ90 Rocket could be found in the open literature. The value of C_{ms} was therefore guessed as 1,8/rad based on the measurements made by *Berner*, *Abate* and *Dupuis* [22].

2.5.8.3.5 Combining the Roll Moments

Using δ_l as calculated from Equation 31 and Mach number M_r of the FZ90 Rocket as calculated in paragraph 2.5.8.5 the overall torque generated by the FZ90 Rocket Body can now be calculated with the **FZ90 Body Torque Estimation** block as illustrated in Figure 67.

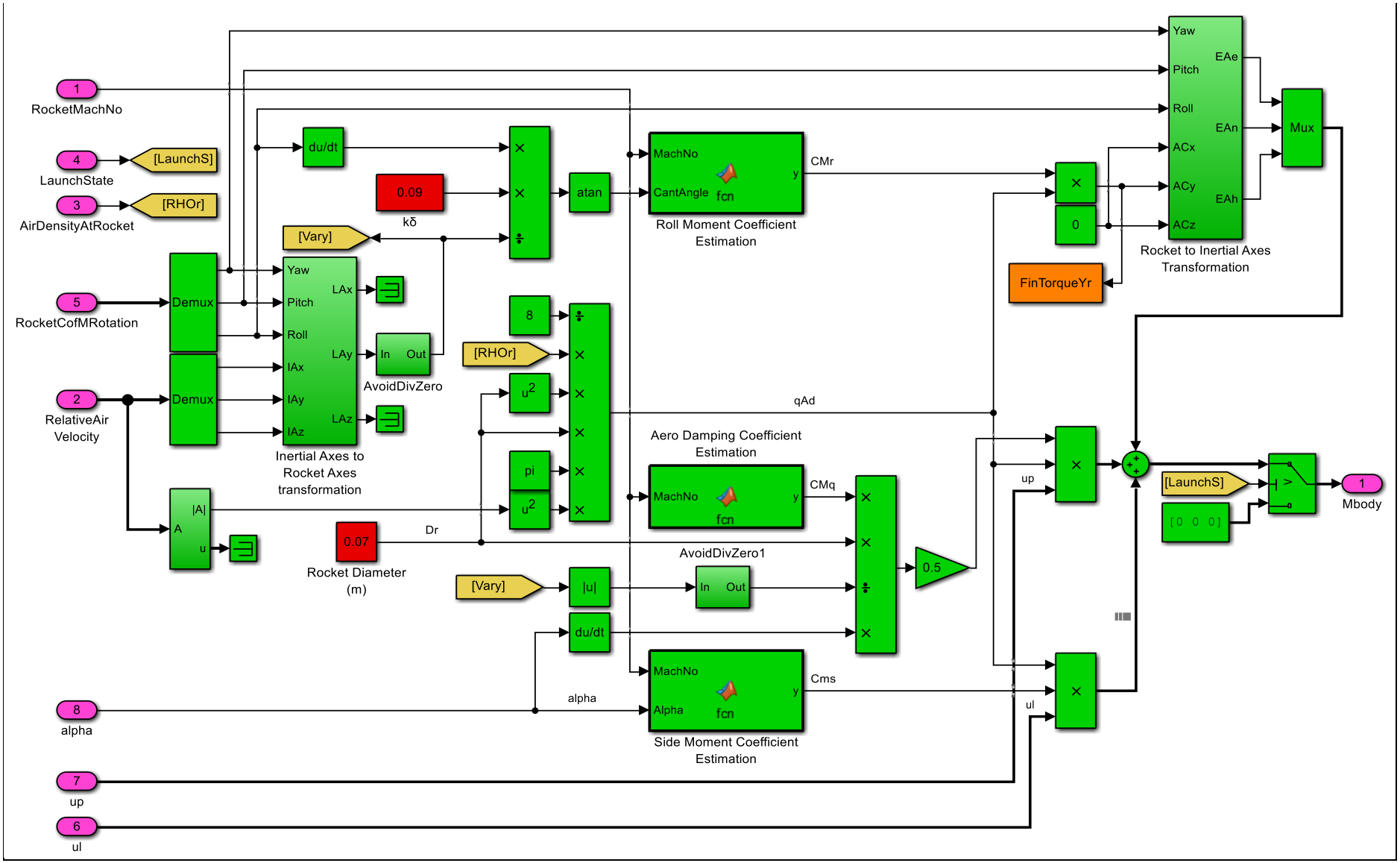


Figure 67. DFBD of the FZ90 Body Torque Estimation block [A Landman, 2017]

Note that Figure 67 makes use of a common dynamic pressure term:

$$qAd = \frac{\rho_r V_{ary}^2 \pi D_r^3}{8} \qquad \textbf{Equation 33}$$

with ρ_r the density of air surrounding the rocket, V_{ar} the magnitude of V_{ar} and D_r the reference diameter (0,07 meter) of the FZ90 Rocket Motor (refer table 5 of Dahlke and Batiuk [21]). The $\frac{\pi D_r^3}{8}$ scaling term in Equation 33 stems from three components: a $\frac{\pi D_r^2}{4}$ term for calculating the cross-sectional area of the FZ90 Rocket, a D_r term to ensure the equation is dimensionally correct (usually the chord of a wing in the case of an aircraft) and a scale factor $\frac{1}{2}$ as required by the dynamic pressure term of the lift equation.

To achieve the real time operation objective of this study the body moment coefficients of the FZ90 Rocket Motor were calculated with separate Matlab Function blocks that can easily be converted into C for precompilation.

Note that the Wrap Around Fin as described in represent a control loop that settles at an equilibrium in spin where the torques acting on the FZ90 Rocket along the **Y**r axis sum to zero. This fact can be used to determine the efficiency scale factor k_δ by simulating the MK-66/M151 trajectory as measured by *Mermagen* and *Oskay* [23]. To do this the *Mermagen* and *Oskay* experiment was emulated by disabling downwash in the **Rocket Launching System Model** (ensuring that LOS perturbations due to downwash are not induced) and configuring the model for the same launch angle (13,5°).

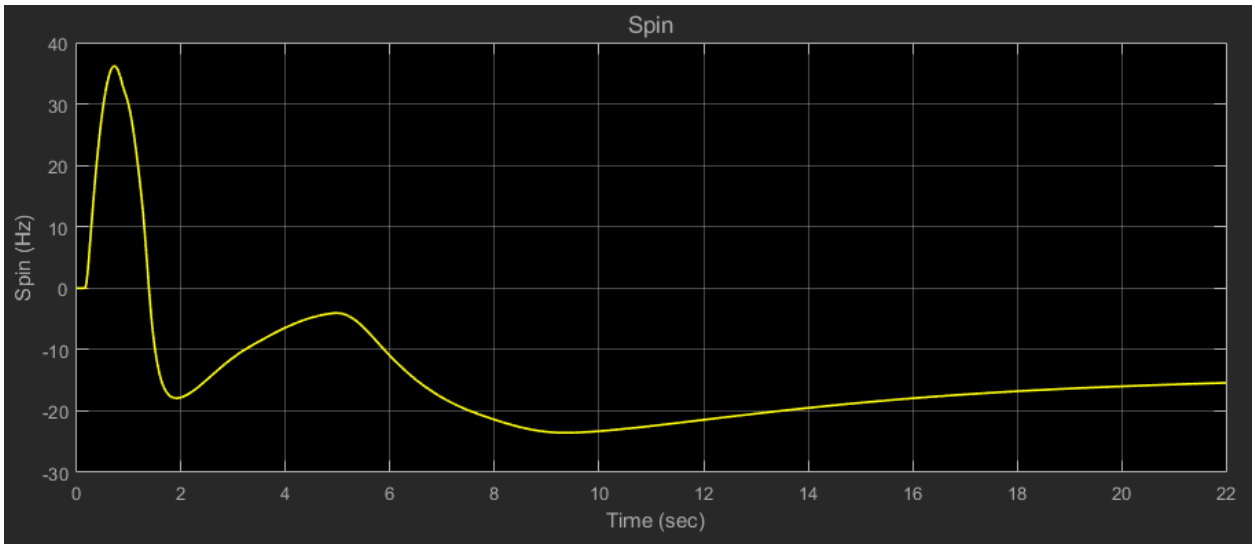


Figure 68. Evolution of FZ90 Rocket spin with k_δ= 0,09 [A Landman, 2017]

Under these conditions the spin of the FZ90 Rocket is mainly determined by the Roll Moment. This reduces the complexity of the problem so that scale factor k_δ can be determined by adjusting it to yield the spin curve as measured by *Mermagen* and *Oskay* [23] and illustrated in Figure 17. The result of such a simulation with k_δ= 0,09 is illustrated in Figure 68 and the correspondence between the two curves is clearly evident.

2.5.8.4 Downwash (Nearfield Flow)

For this version of the **Rocket Launching System Model** the approach followed to model the downwash of Rooivalk AH-2A is similar to that of the ballistic algorithm of the BELL TEXTRON COBRA helicopter as described by *Breaux* [30]. This model represents the downwash as a step function with constant velocity, anti-parallel to the gravity vector and within a radial range of 6,279 meter from the aircraft (i.e. below the main rotor disk) with the magnitude of the downwash selected in the range 3,2 to 13,716 m/s depending on flight conditions.

The values used by the Cobra ballistics model cannot be used here because the Cobra Helicopter has an all-up weight of about 5 ton versus Rooivalk AH-2A which has an all-up weight of about 8 ton. As shown below, this difference causes a sizeable difference in downwash speeds.

To derive values applicable to Rooivalk AH-2A use was made of the rocket equation (i.e. conservation of momentum) as described by *Wertz* and *Larson* [35] as well as *Campbell* and *McCandless* [37]. As first-order approximation, the Author assumed that since the orbital speed of a Main Rotor Blade increases towards the tip of the blade, most of the air pumped by the rotor disk occurs in the outer part of the rotor disk surface. Hence, the following approximation should hold:

$$A_p = k_p A_{mr} \qquad \textbf{Equation 34}$$

with A_p the effective pumping surface of the rotor disk, A_{mr} the overall surface area of the rotor disk and k_p the ratio of the two surface areas.

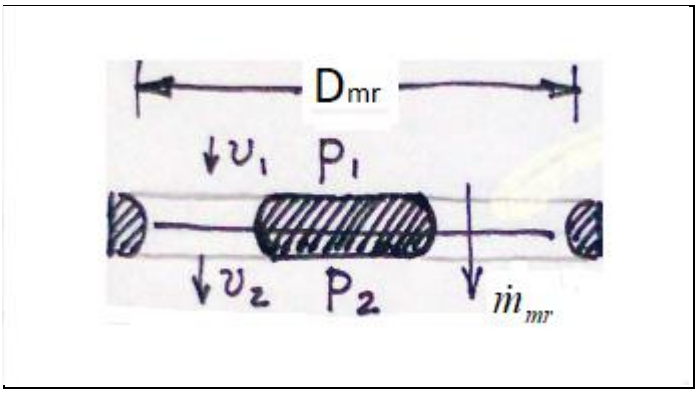


Figure 69. Side-view of main rotor disk modelled as toroidal jet [A Landman, 2017]

Neglecting the effect of vortex flow at the outer edge of the rotor disk[6] by assuming the rotor disk to be a type of toroidal jet as illustrated in Figure 69 and assuming the change in density of the air as it moves through the rotor disk to be negligibly small, application of the rocket equation yields the following relationship:

$$\begin{aligned} F_l &\approx \dot{m}_{mr}\left(v_2 - v_1\right) \\ &\equiv k_p A_{mr} \rho_{mr} v_1 \left(v_2 - v_1\right) \end{aligned} \qquad \textbf{Equation 35}$$

[6] This is a reasonable first-order approximation because the Main Rotor Blade used on Rooivalk CSH has an Aspect Ratio of about 12. The effect of vortex flow at the tips of the rotor blades is to cause induced drag which reduces the effectivity of the rotor disk. However, the effect decreases with increasing Aspect Ratio, and can probably be neglected at a value of 12 for the purposes here. The model would predict even higher downwash velocities and air mass flow rates should the effect be taken into account.

with F_l the lift generated by the Main Rotor, v_1 the inflow velocity, v_2 the outflow velocity and ρ_{mr} the air density above the Main Rotor. Assuming the gravitational constant to be 9,8 m/s^2 the following must hold for a hovering helicopter:

$$\frac{k_p \pi D_{mr}^2 \rho_{mr} v_1 (v_2 - v_1)}{4} \approx 9{,}8 M_{ac}$$

$$\Rightarrow \quad v_2 \approx \frac{39{,}2 M_{ac}}{k_p \pi D_{mr}^2 \rho_{mr} v_1} + v_1$$

Equation 36

with D_{mr} the diameter of the Main Rotor and M_{ac} the mass of the Aircraft. Equation 36 demonstrates an important principle of a helicopter main rotor that will form one of the main pillars of this study - outflow velocity is dependant on the inflow velocity. For very small inflow velocities v_2 is almost entirely inversely proportional to v_1 so that very high outflow velocities are required to generate the lift required to keep the aircraft airborne. In other words, there is insufficient fluid to accelerate through the Main Rotor. Such a main rotor would be inefficient because it is choked. However, as the inflow velocity is increased the massflow rate increases and the difference between inflow and outflow velocities reaches a minimum.

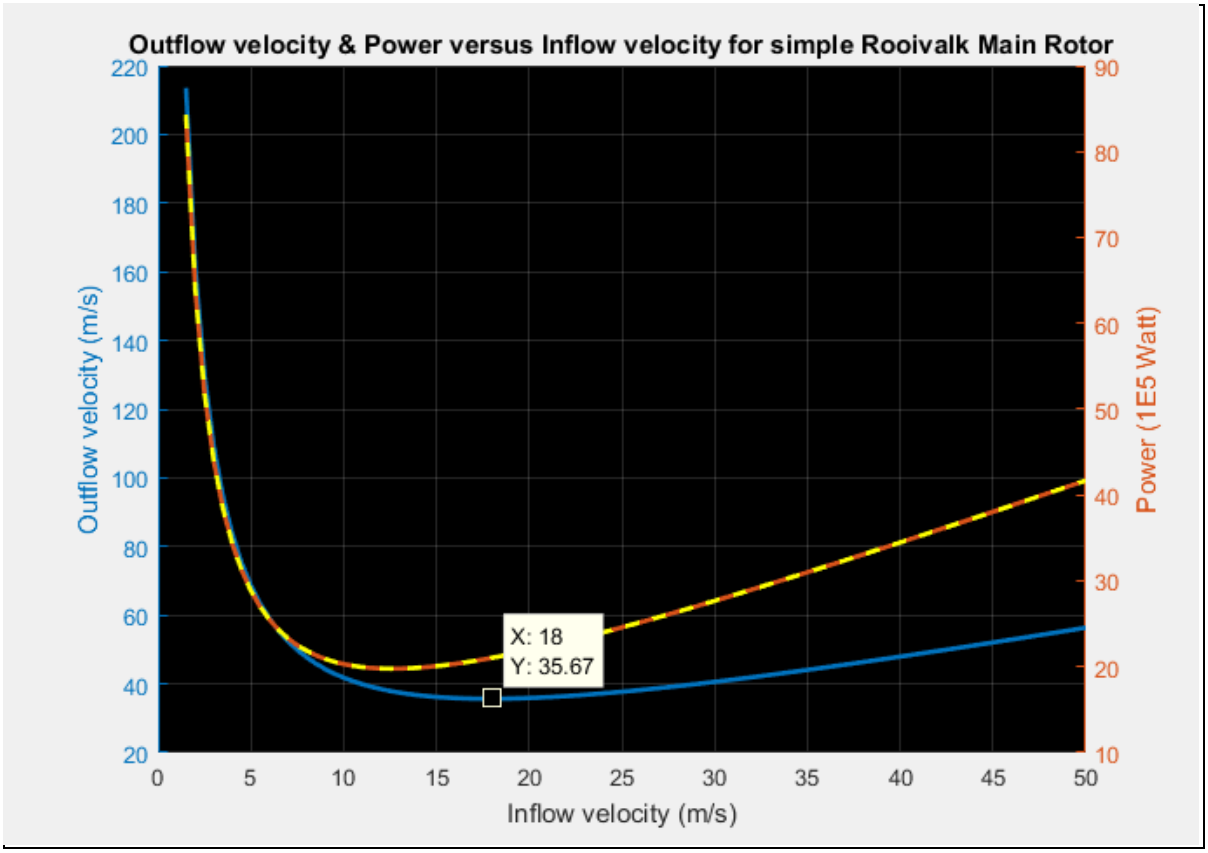


Figure 70. Outflow versus Inflow velocities for simple Rooivalk Main Rotor model [A Landman, 2018]

Applying Equation 36 the mass flow rate $\dot{m}_{mr}$ through the main rotor is given by:

$$\dot{m}_{mr} = \frac{k_p \pi D_{mr}^2 \rho_{mr} v_1}{4}$$

Equation 37

The mechanical power P_h required from the engines to drive the Main Rotor is given by:

$$P_h \approx \frac{1}{2}\dot{m}_{mr}\left(v_2^2 - v_1^2\right) \qquad \textbf{Equation 38}$$

To apply Equation 36 and Equation 38 to Rooivalk AH-2A note that the density of air at 0°C and 1 Atmosphere is 1,2929 kg/m^3 while the diameter of the Main Rotor of Rooivalk AH-2A is 15,580 meter. Assuming Rooivalk AH-2A to have an all-up mass of 8000 kg with k_p = 1,0 the relationships as illustrated in Figure 70 results. It is evident that an optimum inflow occurs at 18 m/s which yields an outflow velocity of 35,7 m/s. At that point the massflow rate is about 4,3 ton/sec.

The implication of this result is significant, because the minimum outflow velocity of 35,7 m/s is comparable to the velocity of 43,6 m/s of the FZ90 Rocket when it leaves the Launcher (see paragraph 2.5.7.1.1). As a result, the FZ90 Rocket is bound to be sensitive to the main rotor downwash of Rooivalk AH-2A as observed. It is also substantially larger than the maximum of 13,716 m/s used by the ballistic algorithm of the BELL TEXTRON COBRA helicopter as described by *Breaux* [30]. Perturbation of the rocket trajectory will therefore be larger when launched from an 8 ton helicopter than from a 5 ton helicopter. It is evident from Equation 36 that the velocity of air in the downwash will increase if the air becomes warmer, so perturbation of rocket trajectory will be worse under hot conditions than under cold conditions.

Also evident from Figure 70 is that the energy required for Rooivalk AH-2A to operate at minimum outflow velocity will be about 2 MW. This is less than the 2,4 MW that can be continuously produced by two Makila Engines as used on Rooivalk AH-2A. Yet it is known that Rooivalk AH-2A can operate on one engine in forward flight. As explanation of this fact note that the main rotor downwash has the shape of a doughnut with air re-cycling through the main rotor so that v_1 might actually be close to 18 m/s during hover. During forward flight the inflow velocity decreases because the Main Rotor (slightly tilted) flies into stationary air. This causes a reduction in the value of v_1 which in turn results in a decrease in power required to fly – as observed on a real helicopter.

It is clear that the model of main rotor and concomitant downwash as developed in this chapter is an over-simplification of a complex process that strongly affects the trajectory of the FZ90 Rocket. As a result the errors introduced in the ballistic estimation process will be large, so the sophistication of the Main Rotor Model and Downwash Model must be suitably increased (the objectives of Chapter 3 and Chapter4).

2.5.8.5 Atmospheric Conditions (Far Field Flow)

For this version of the **Rocket Launching System Model** the approach followed to model the atmosphere (i.e. conditions far from the helicopter where effects of downwash are negligibly small) is to assume the standard atmosphere as specified by ICAO [51].

For the purposes of this study the equations as described by Cavcar [52] were used to represent the characteristics of the ICAO standard atmosphere. Air temperature is assumed to decrease with height above ground as follows:

$$T_a = T_{ao} - 0{,}0065\, y_a \qquad \textbf{Equation 39}$$

with T_a the air temperature (in °Kelvin) at height above ground y_a (in meter) and T_{ao} the air temperature (in °Kelvin) at ground level. The speed of sound is given by:

$$V_s = 20{,}04680\sqrt{T_a}$$ **Equation 40**

with V_s the speed of sound in m/s. This allows for the calculation of the Mach number of a body as the ratio of the relative air velocity of the body to V_s. The density of air at height above ground y_a is given by the following:

$$\rho_a = \rho_{ao}\left(\frac{T_a}{T_{ao}}\right)^{4,25588}$$ **Equation 41**

with ρ_a the density of air at height above ground y_a and ρ_{ao} the standard density of air at ground level. T_a represents the temperature of the air at height above ground y_a and T_{ao} the temperature of the air at ground level.

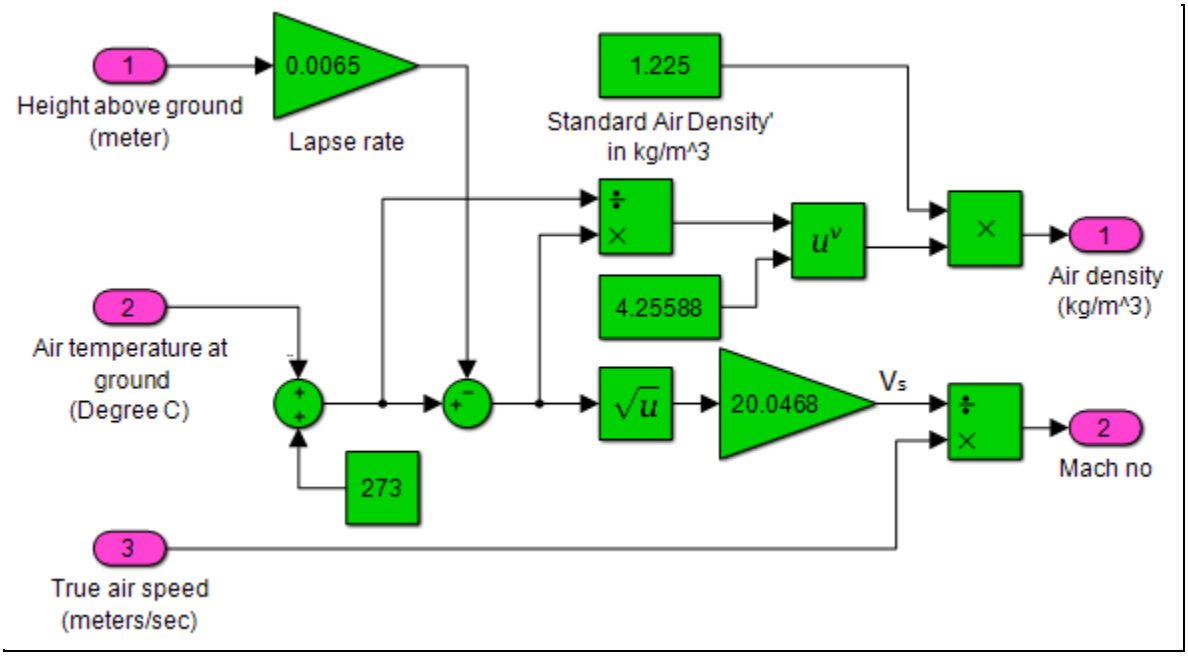


Figure 71. DFBD for calculating atmospheric conditions at rocket [A. Landman, 2014]

The DFBD for calculating atmospheric conditions at the rocket using Equation 39, Equation 40 and Equation 41 is illustrated in Figure 71.

2.5.9 Simulating the RLS of Rooivalk AH-2A

To simulate the Rocket Launching System of Rooivalk AH-2A the **Rocket Launching System Model** as illustrated in Figure 20 was configured with the blocks as described in paragraphs 2.5.6, 2.5.7 and 2.5.8. All the other blocks were blanked out since the objective here was only to model the coning behaviour of the FZ90 Rocket as observed. To test the predictions made in paragraph 2.4 a number of loadcases were run as described below.

2.5.9.1 Loadcase 1

This loadcase simulates the *Mermagen* and *Oskay* [23] launch experiment using the FZ90 Rocket instead of the MK-66 Rocket. The model was configured as follows:

a) Aircraft stationary.
b) Launch from Weapon Station 2 (i.e. second from Pilot's left).
c) Launcher yaw angle at +0° (i.e. pointing towards **Y**i-axis).
d) Launcher pitch angle at +13,5°.
e) Launcher roll angle at +0° (i.e. Aircraft in level attitude).
f) Rocket CoM at +3 m above origin.
g) No rotor downwash.
h) No ambient wind.
i) Ambient air temperature +11°C.
j) Experiment done at sea level.
k) Rocket bulk temperature +22°C.
l) Trigger Pulse generated after 0,15 second (to allow FZ90 Rocket to settle in Launch Tube).

The simulated performance of the RLS system of Rooivalk AH-2A under Loadcase 1 is illustrated in Figure 72. Comparing the results with those as measured by *Mermagen* and *Oskay* [23] (see) using a MK-66 Rocket launched under similar conditions it is evident that the **Rocket Launching System Model** produces very similar results, especially in spin (which was the objective of the *Mermagen* and *Oskay* [23] experiment). Some of the main features are compared in Table 5.

Table 5. Comparison: Mermagen and Oskay versus RLS Model [A Landman, 2018]

Characteristic	Mermagen and Oskay	Rocket Launching System Model
Maximum rocket velocity	680 m/s	720 m/s
Maximum rocket spin	35,0 Hz	35,5 Hz
First spin minima after spin reversal	18,0 Hz	18,4 Hz
Time to First spin minima after spin reversal	2,0 sec	2,0 sec
Closest spin to forbidden zone after spin reversal	4,3 Hz	5,4 Hz
Spin at ground impact	15,3 Hz	14,4 Hz

Hence, we conclude that the **Rocket Launching System Model**, with the exclusion of downwash effects, can be considered as accurate within a few percent.

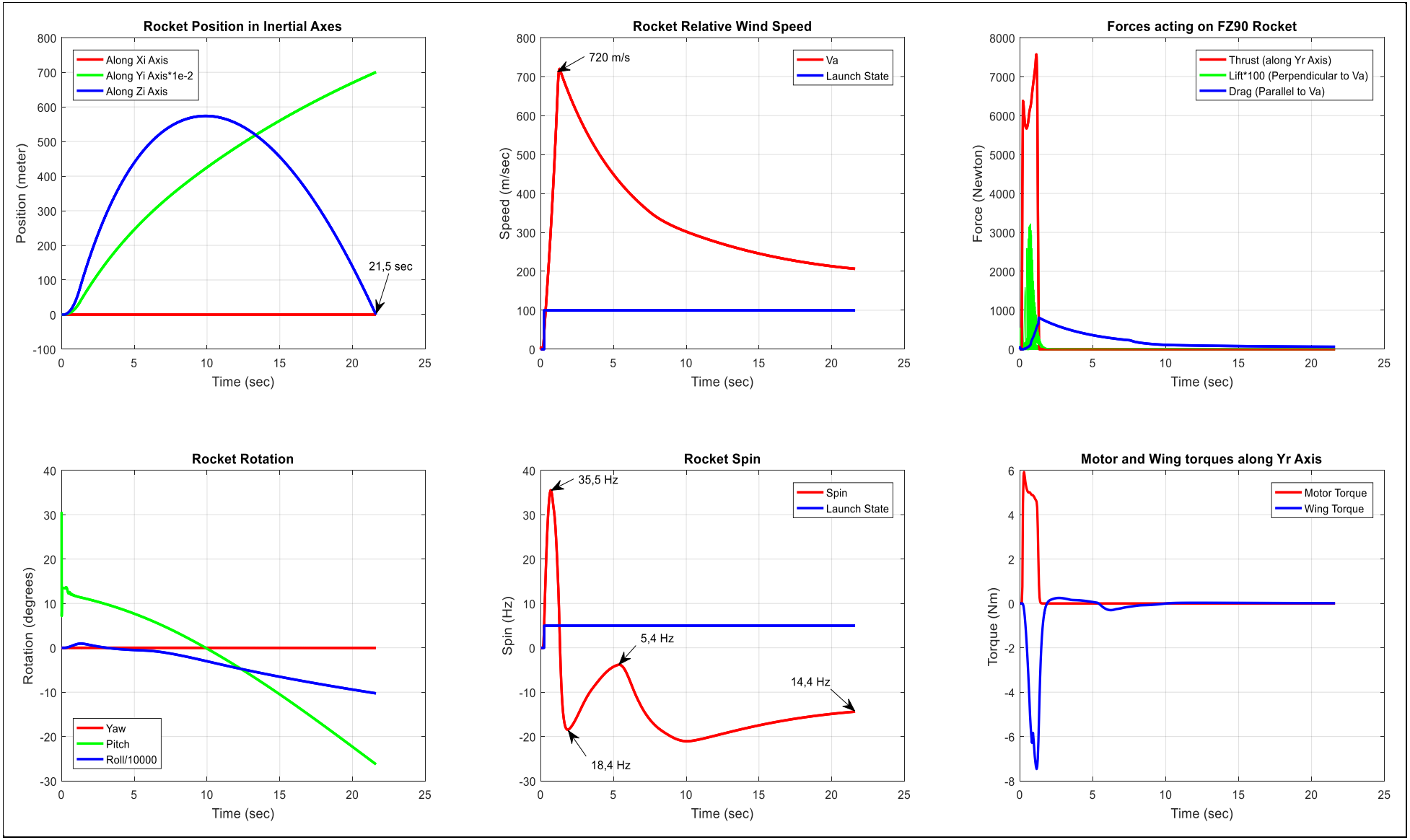


Figure 72. Evolution of some FZ90 Rocket characteristics during Loadcase 1 [A Landman, 2017]

2.5.9.2 Loadcase 2

This loadcase simulates the *Mermagen* and *Oskay* [23] launch experiment in the presense of downwash using the FZ90 Rocket instead of the MK-66 Rocket. The **Rocket Launching System Model** was configured as follows:

a) Aircraft stationary.
b) Launch from Weapon Station 2 (i.e. second from Pilot's left).
c) Launcher yaw angle at +0° (i.e. pointing towards **Y**i-axis).
d) Launcher pitch angle at +13,5°.
e) Launcher roll angle at +0° (i.e. Aircraft in level attitude).
f) Rocket CoM at +3 m above origin.
g) Downwash with aircraft in hover and *Horizontal Downwash Gain* set to 3.
h) No ambient wind.
i) Ambient air temperature +11°C.
j) Experiment done at sea level.
k) Rocket bulk temperature +22°C.
l) Trigger Pulse generated after 0,15 second (to allow FZ90 Rocket to settle in Launch Tube).

The simulated performance of the RLS system of Rooivalk AH-2A under Loadcase 2 is illustrated in Figure 73. It is evident that the FZ90 Rocket is very sensitive to downwash in both yaw and pitch, causing changes in every parameter that describe the launch process.

Inspection of Figure 73 shows that the downwash triggers a sinusoidal disturbance in rocket pitch which decays in an exponential fashion similar to that as observed in Figure 15. The disturbance is established within about 0,1 second, starting from 13,5° as the FZ90 Rocket leaves the Launcher to 32,6° a short time after it leaves the downwash area. The end result is a residual shift of 5° in rocket pitch that changes the vertical trajectory of the rocket as illustrated at top left of Figure 73.

Similarly, the downwash triggers a sinusoidal disturbance in yaw which decays in an exponential fashion similar to that of pitch. This disturbance starts from 0° as the FZ90 Rocket leaves the Launcher to 8,3° a short time after it leaves the downwash area. The end result is a residual shift of 2° in yaw that causes a lateral drift of 304 meter (not shown here) in the trajectory of the rocket at ground impact (assuming a flat earth).

The disturbances in yaw and pitch causes the Angle of Attack of the FZ90 Rocket to evolve as illustrated at bottom left of Figure 73. A rapid increase in Angle of Attack to a maximum of 29,5° over a period of about 30 msec followed by a rapid decrease to about 6° generates an input that effectively represents an impulse. The sinusoidal disturbances with exponential decay in yaw and pitch as illustrated by the graphs at centre of Figure 73 represent the response of the FZ90 Rocket to this impulse. As shown, it is typical of a second-order system except that it does not return to its initial condition because the rocket motor is in Boost stage while the disturbance occurs, thus imparting velocity components along un-wanted directions.

Inspection of top centre graph of Figure 73 shows that the maximum perturbation angle achieved during Loadcase 2 is about of 27,6° in pitch and 8,3° in yaw. Initially these disturbances are in phase (polarized motion) which then gradually shift to establish a 90° shift (circular motion) after about 2 seconds. Hence, the initial large movement of Figure 18 is more of a skewed gallop (aligned with the direction of the downwash) than a conical motion. This large initial movement gradually morphs into a coning motion with precession angle of about 1,1°. This strange motion can be visualized by plotting the Zi and Xi coordinates of unit vector **ur** against time as illustrated in Figure 74. This same motion can also be observed in Figure 13.

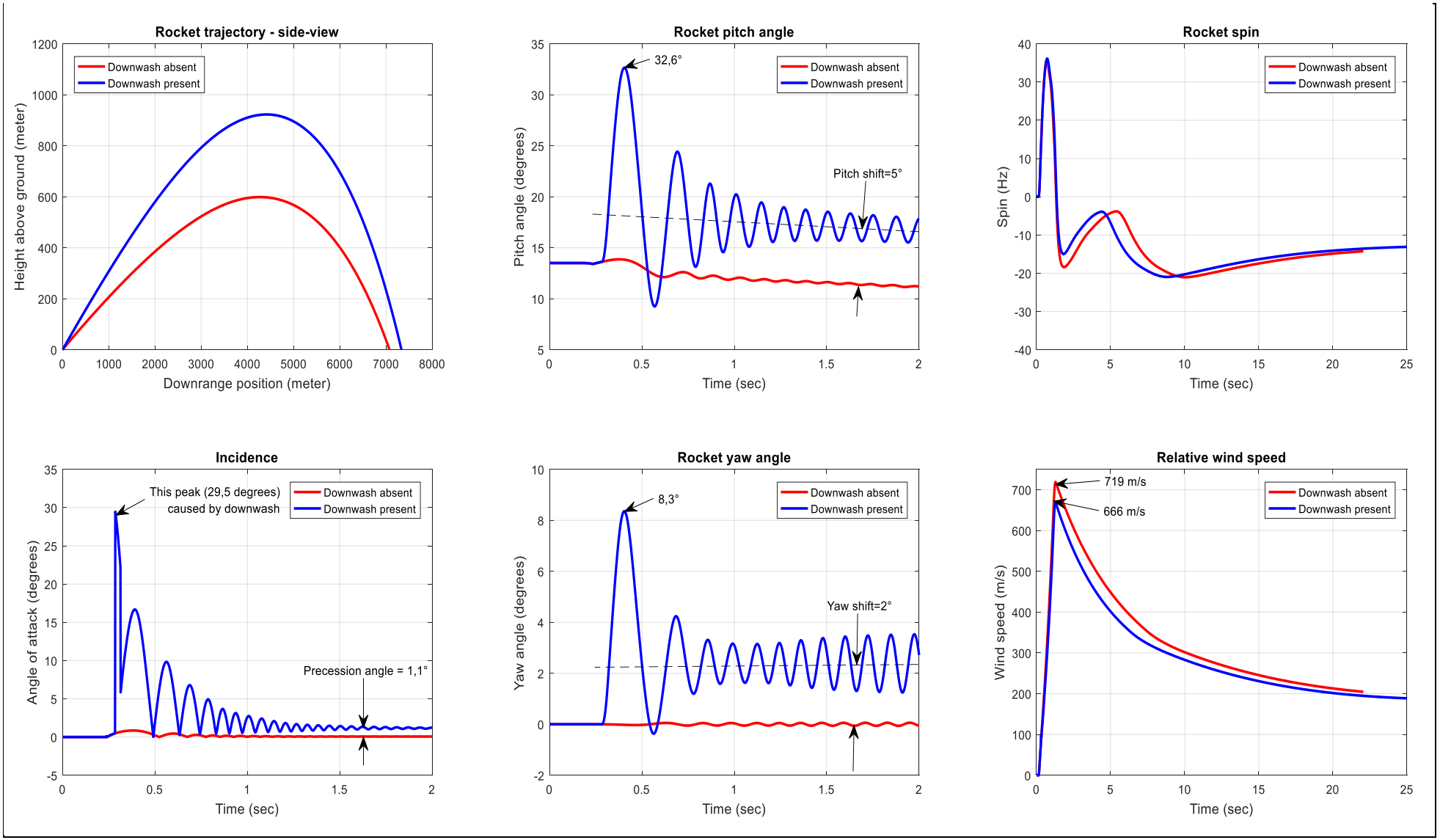


Figure 73. Evolution of some FZ90 Rocket characteristics during Loadcase 2 [A Landman, 2017]

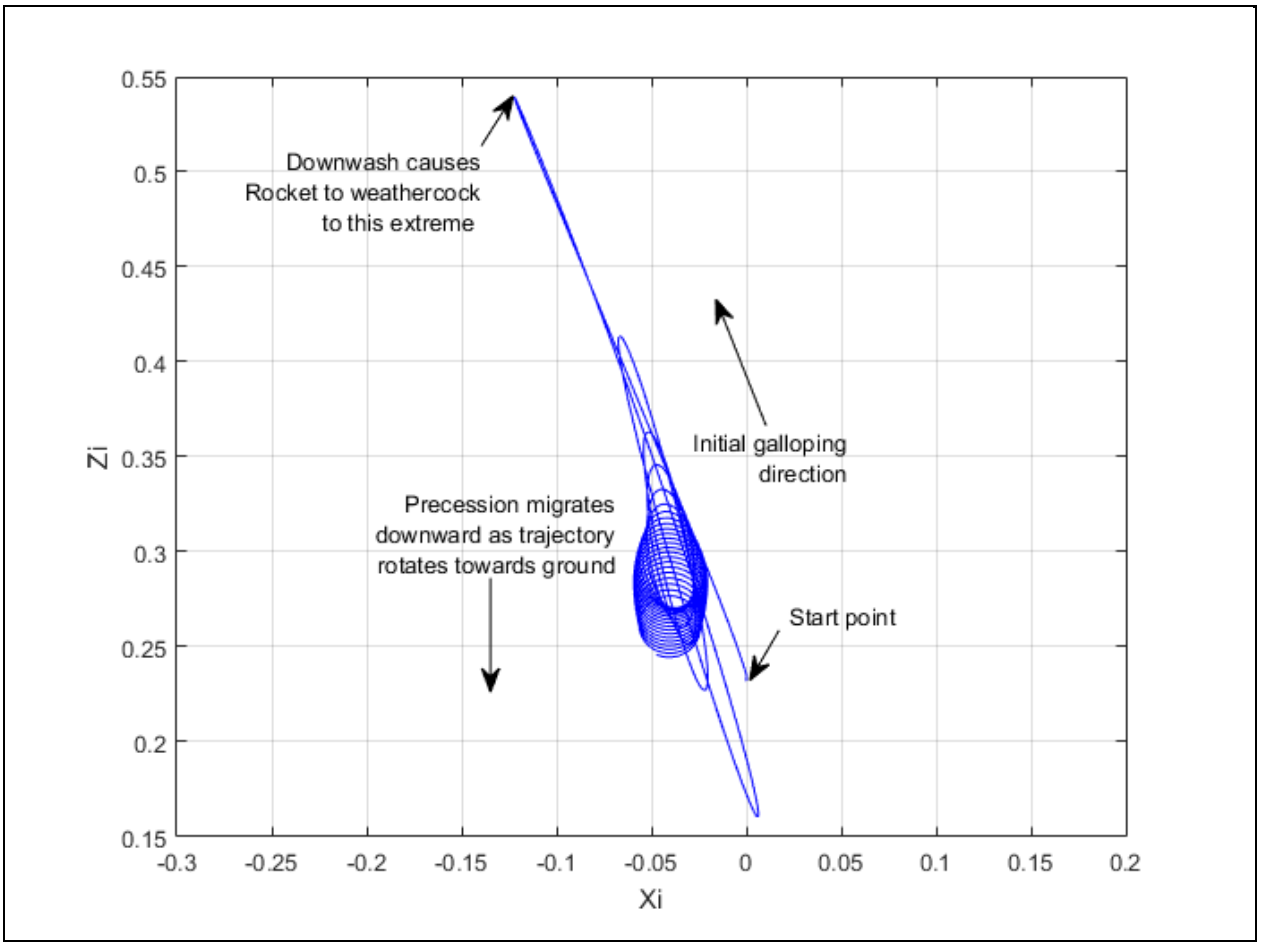


Figure 74. Locus traced by unit vector UR on XZ plane during Loadcase 2 [A Landman, 2018]

One of the effects evident from bottom centre of Figure 73 is droop of the FZ90 Rocket directly after leaving the Laucher when launched in the absence of downwash. This effect causes the pitch angle of the FZ90 Rocket to drop with a few degrees over a period of 0,5 second flight time. Downwash causes the opposite but with much larger effect as confirmed by the work of *Lee*, *Choi* and *Kwon* [28]. Both of these effects have some effect on both crossrange and downrange distance, especially at low launcher pitch angles as simulated here. While their combined effect could be calculated here, the open literature has little to say about the variation in impact point (ballistic accuracy) of the FZ90 Rocket under these conditions, other than the simplistic 12 milliradians/1000 meter assumption of *Wonnacott* [9] for the MK-66 Rocket.

Finally, inspection of the graphs on right of Figure 73 shows downwash does not have a very strong influence on spin and speed of the FZ90 Rocket. The reason for this small influence is evident from Figure 75 and Figure 76 which show that the forces and torques acting on the FZ90 Rocket are not affected much by downwash. It is evident that downwash indirectly affects the trajectory of the FZ90 Rocket through changes in pitch and yaw during the Boost stage (which are easy because the FZ90 has such a big stability margin). Most importantly, although downwash does cause a delay in the evolution of rocket spin it does not affect the basic shape nor the safety limits generated by the bevels and tabs of the Wrap Around Fin.

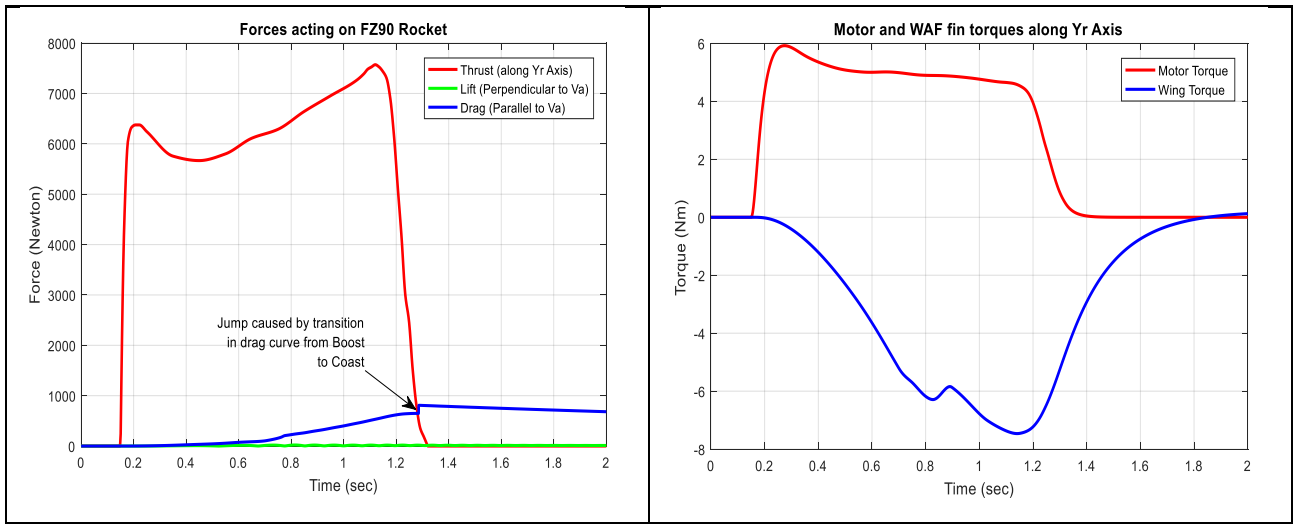


Figure 75. Forces and torques acting on FZ90 Rocket during Loadcase1 [A Landman, 2014]

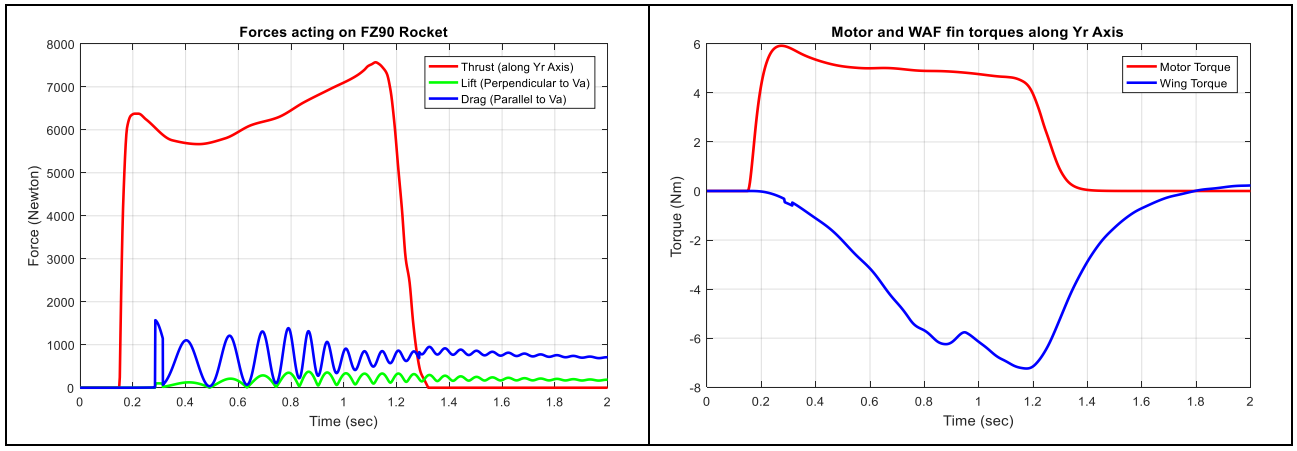


Figure 76. Forces and torques acting on FZ90 Rocket during Loadcase 2 [A Landman, 2014]

2.5.9.3 Loadcase3

This loadcase was an attempt to emulate the coning conditions as observed in Figure 15. Hence, the model was configured to represent an aircraft at heigh of 200 ft Above Ground Level (AGL) with Rocket Launcher at elevation angle such as to yield a downrange distance of 2000 meter. To do this the **Rocket Launching System Model** was configured as follows

a) Aircraft stationary.
b) Launcher yaw angle at +0° (i.e. pointing towards **Y**i-axis).
c) Launcher pitch angle at -4,4° (to yield 2000m downrange distance).
d) Launcher roll angle at +0° (i.e. Aircraft in level attitude).
e) Rocket CoM at 200 ft (i.e.61 meter) above origin.
f) Rotor downwash present.
g) No ambient wind.
h) Ambient air temperature +15°C.
i) Experiment done at sea level.
j) Rocket bulk temperature +22°C.
k) System held steady for 0,1 second (to allow FZ90 Rocket to settle in Launch Tube) after which the Trigger Pulse is generated.

Note that under these conditions the (upward) pitch shift induced by the downwash has to be countered with a Launcher pitch angle of -4,4° for the FZ90 Rocket to reach the ground at 2000 meter range.

In interpreting the behaviour of the FZ90 Rockets as observed in Figure 15 an important factor to consider is that the Boost phase is characterized by two distinct stages. The first stage (K2SO4 burn) is demarcated by consumption of a K_2SO_4 rod integral to the MK-66 MOD4 Rocket as described by *Hawley* [11]. The purpose of the rod is to generate surplus oxygen, ensuring complete combustion within the combustion chamber to avoid flame-outs of the helicopter engines early during the Boost phase. The second stage (Normal burn) starts when the K_2SO_4 rod has been consumed. It is characterized by a secondary combustion flame some distance (2-3 rocket lengths) behind the FZ90 Rocket as can be seen in Figure 16. This flame appears red during the night, thus producing the galloping pattern of Figure 15 as a result of the movement of the FZ90 Rocket.

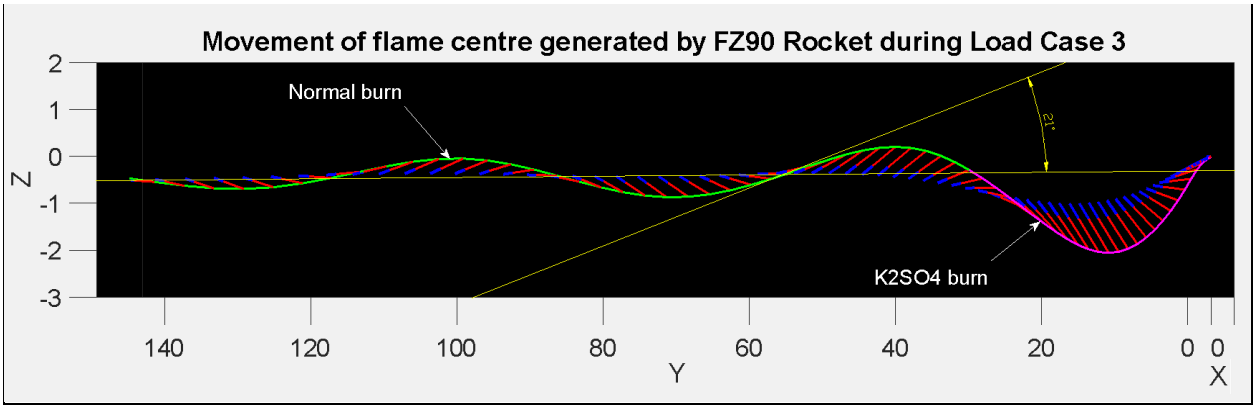


Figure 77. Movement of flame centre during Loadcase3 [A Landman, 2018]

The appearance of the galloping pattern produced by the FZ90 Rocket and viewed from a position to the rear left of the helicopter was emulated with the **Rocket Launching System Model** configured as described above. The result is illustrated in Figure 77, showing the perspective view from a position that produced a “coning angle” of 21° as derived from Figure 18.

2.6 Summary (Develop General Theories)

This chapter was used to initiate the second Scientific Method Cycle of a family of nested Scientific Method Cycles that yield the concept of a Precision Rocket Launching System. To implement the PRLS concept the error mechanisms resulting in the poor ballistic performance of standard Free Flight Rockets (FFR) must be understood. The purpose of this chapter was to investigate and model one of the primary error mechanisms of dumb rockets referred to as the **Coning Problem.**

The **Coning Problem** was described by presenting as main observation evidence photographs of the coning and corkscrew behaviour of salvos of FZ90 Rockets launched from Rooivalk AH-2A. Additional observations from a range of researchers in the field was also presented.

Probing questions were asked and used to formulate the Coning Hypothesis which postulates that the coning and corkscrew behaviour of the FZ90 Rocket is the joint interaction between rocket flight dynamics and downwash flow field in vicinity of Rooivalk AH-2A, amplified by the butterfly effect. The Coning Hypothesis also postulates that the shape of the downwash flow field is in the form of a doughnut that elongates towards the rear of the Aircraft during forward flight. At 80 knots, the downwash bends over the Stub Wings, causing the FZ90 Rockets to be launched into undisturbed air which avoids the error caused by coning (but other error sources to remain).

A number of predictions based on the Coning Hypothesis were made. Some of these predictions were based on data from the photographic evidence and some were based on data from the open literature.

To gather data for testing the Coning Hypothesis a multibody model written in Simulink and referred to as the the **Rocket Launching System Model** was developed using studies and data from a wide range of researchers in the field. This model was configured to model the joint behaviour of M159 Rocket Launcher, FZ90 Rocket, Downwash of Rooivalk AH-2A and Atmosphere. The simulation results show very good agreement with observational data and confirms the assumptions made in formulating the Coning Hypothesis.

Having performed the research of Chapter 2 the following conclusions are now formulated:

a) The FZ90 Rocket is very sensitive to downwash. To implement the Model Predictive Control Loop as hypothesized in paragraph 1.3 the downwash flow pattern will have to be calculated in much detail and in real time.

b) The pressure and shear force distribution over a wing (or lifting body) as described by *Anderson* [45] must be known for at least 10 equal areas of influence over the length of the FZ90 Rocket body.

c) The flat-disk model of main rotor and concomitant homogenous downwash as developed in this chapter is an over-simplification of a complex process that strongly affects the trajectory of the FZ90 Rocket. To achieve the PRLS Hypothesis the sophistication of the Main Rotor Model and Downwash Model would have to be substantially increased (this becomes the objectives of Chapter 3 and Chapter4).

d) The **Rocket Launching System Model** as developed in Chapter 2 is sufficient (with some refinements) to represent the **RLS Model** block of the Model Predictive Control Loop as hypothesized in paragraph 1.3, provided the actual characteristics of the particular rocket and aircraft used are known to a high degree of accuracy. There can be no uncertainty between truths and half-truths as encountered in Chapter 2.

e) The error caused by downwash is the primary error source but by no means the only one. Other error sources such as bending of the Stub Wings during launch, thrust mis-alignment and rocket wobble must also be considered in real time for the Model Predictive Control Loop as hypothesized in paragraph 1.3 to work.

It would appear that the open literature has either ignored or under-estimated the effects of downwash. All the papers cited with the exception of *Lee*, *Choi* and *Kwon* [28] and *Hawley* [12] consider only particular aspects of rockets that can be evaluated in a wind tunnel – they don't evaluate the rocket as part of a bigger RLS system. This makes many of their conclusions subject to re-interpretation or not applicable to rockets launched from a helicopter.

The first contribution of this chapter is explanation of perturbation of the rocket trajectory due to downwash as one of the primary error sources of rockets launched from a helicopter. The second contribution is the fact that for a given inflow velocity the outflow velocity of a main rotor is defined. As a result, the main rotor of a helicopter reacts like a current controlled current source in electrical circuit parlour. This makes it possible to analyse in isolation or as a whole the flow of a combined system comprising a Main Rotor and Space (i.e. the air around the Helicopter).

3 The Main Rotor Model (Cycle 3)

"*To invent, you need a good imagination and a pile of junk*". Thomas Edison

This chapter introduces the third Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. As shown in Chapter 2 the downwash of the helicopter must be calculated with sufficient (not perfect) accuracy and in real time for the PRLS concept to work. Although very simple and easy to run in real time the toroidal jet model of the main rotor and concomitant homogenous downwash as developed in Chapter 2 was found to be an over-simplification of a complex process. In reality this process involves the interaction of many components including Space, Main Rotor Blades, Main Rotor Hub, Swash Plate, Flapping Hinges and Engines as illustrated in Figure 78. Although the toroidal jet model was an over-simplification, it led to the conclusion that the main rotor reacts like a 3-dimensional current controlled current source in electronic parlour. This concept will be refined in this chapter.

Figure 78. The main downwash-inducing agents: Main Rotor Hub, Blades, Engines and Fuselage [Gary Shephard, 2008]

The purpose of this chapter is to conceive the **Main Rotor** block of the **Rocket Launching System Model** as illustrated in Figure 20, such that it is (just) good enough to achieve the PRLS accuracy as postulated in Chapter 2. To achieve such an objective at the current stage of this study is strictly not possible, since **good enough** is ultimately an expression of overall PRLS performance. For the purposes of this Scientific Method Cycle the **good enough** objective is persued by increasing the sophistication of the model in steps, evaluating the performance of the resulting models against real world values and making judgement based on intuition as described by various researchers such as *Kokalari et al* [28], *Woodson & Melcher* [29] and *Karnopp, Margolis & Rosenberg* [31].

The primary observational evidence presented for developing the **Main Rotor Model** includes the mechanical layout of the Main Rotor as used on Rooivalk AH-2A, the Actuator Disk Theory of *Froude* [53], the rocket equation, the equations for aerodynamic lift and drag and the fact that viewed sideways the main rotor of a helicopter seems to form a virtual cone which changes shape depending on what the pilot does with his controls. It also includes concepts from electrical circuit analysis such as transmission lines (in which disturbances influence each other and travel with finite speed) as well as current sources (which source constant currents through their terminals irrespective of the voltages present at these terminals).

Note: *The term "current source" here refers to current produced by a current source irrespective of voltage at terminals, analog to shape produced by a rigid body irrespective of pressure distribution over the surface of the rigid body. This follows the assertion by Anderson [45] that aerodynamic effects such as lift and drag are caused by a body causing a disturbance of space (current) which generates a pressure and shear distribution (voltage) over the surface of the body in response.*

Questions are asked as to what type of model would satisfy the requirement to model the main rotor of a helicopter in real time and with sufficient accuracy. From the research performed in Chapter 2 it is evident that the FZ90 Rocket is very sensitive to downwash. For this reason the **Main Rotor Model** must be such that it can excite both large and small flows of the downwash flow pattern that can have an influence on the trajectory of the FZ90 Rocket.

Questions are asked as to what approach researchers like *Froude* [53] and *Chollet* [54] followed to model propellers and main rotors of helicopters in the past This leads to the Floppy Disk Hypothesis which states that the sophistication required to model with good enough fidelity the flow through the main rotor of a helicopter for the purposes of the PRLS can be achieved by representing the main rotor as a thin 3-dimensional surface of current controlled current sources (jets) which morphs into various shapes and velocity distributions as a function of flight conditions and flight control inputs.

A model of the main rotor of a helicopter is subsequently developed by calculating the balance of forces which defines the (floppy) shape of the main rotor. The blades are divided into small annular sections as originally described by *Froude* [53] such that the sections sweep a fine array of control surfaces (orifices) over the surface of the main rotor disk. This allows application of a force balance at each control surface, using a combination of the rocket equation and the standard equations for aerodynamic lift and drag. The resulting **Main Rotor Model** is then used to perform predictions of the flow distribution over the rotor disk surface of Rooivalk AH-2A and compared with measured values.

The main contributions of this chapter is development of the **Main Rotor Model** based on a combination of Blade Element Theory, the rocket equation and standard theory of aerodynamic forces of lift and drag. In conceiving the **Main Rotor Model** we show that such a model cannot be used in isolation, but that it has to be integrated with the **Space** block of Figure 20 as a minimum.

3.1 The Main Rotor of Rooivalk (Make observations)

The Main Rotor of Rooivalk AH-2A is fully articulated and uses four blades as illustrated in Figure 79. It has a diameter of 15,58 meter and rotates at nominal rate of 265 rotations/minute in a clockwise direction as evident from the orientation of blades illustrated in Figure 79. Each blade has a cord length of 600 mm, comprises four sections of the SA131 aerofoil family and has a twist of -1,548°/meter.

Figure 79. The Main Rotor of Rooivalk AH-2A [A Landman, 2018]

3.2 How to model the Main Rotor? (Think of interesting questions)

The **Toroidal Jet Model** of the Main Rotor and concomitant homogenous downwash as developed in Chapter 2 was found to be an over-simplification of a seemingly complex process that strongly affects the trajectory of the FZ90 Rocket. So how should the downwash be modelled to yield enough detail for calculating the trajectory of the FZ90 Rocket with **good enough** accuracy to satisfy the PRLS Hypothesis?

To answer this question with certainty at the current stage of this study is not possible, since in this case **good enough** is ultimately a statement of overall PRLS performance – the ultimate calculation of which lies beyond this study as described in Chapter 8. For the purposes of this Scientific Method Cycle the **good enough** objective will therefore be pursued by increasing the sophistication of the model in steps, evaluating the performance of the resulting models against real world values and making judgements based on facts, intuition and experience as suggested by various researchers such as *Kokalari et al* [33], *Woodson & Melcher* [29] and *Karnopp, Margolis & Rosenberg* [31].

From the research performed in Chapter 2 it is evident that the FZ90 Rocket is very sensitive to downwash along both the $\boldsymbol{Z}_r$ and $\boldsymbol{X}_r$ axes. For this reason the **Main Rotor Model** must be such that it can excite both large and small flows of the downwash flow pattern that can have an influence on the trajectory of the FZ90 Rocket. Experience and gut-feel would seem to indicate that the size of these local flows should be an order of magnitude smaller in size than the length of the FZ90 Rocket. In addition, the flow pattern over the Main Rotor Disk should be modelled as a 3-dimensional flow field versus the 2-dimensional flow field as in Blade Element Theory originally formulated by *Froude* [53]. Note that this conclusion also means that objects such as fuselage, ground and buildings need to be considered in calculating downwash (this is one of the objectives of Chapters 4 and 5).

Finally, note that in paragraph 2.5.8.4 the lift produced by the main rotor of a helicopter was predicted with the rocket equation. Since a main rotor is simply a number of wings flying in a circle, the forces produced by a main rotor are the standard aerodynamic forces of lift and drag.

Ultimately, the rocket equation and the equations for aerodynamic lift and drag describe the same thing. This equivalence will be referred to as the **Equivalence of Force Principle** for the purposes of this study.

3.3 The Flexible Disk Hypothesis (Formulate hypothesis)

Considering the observational evidence we now postulate that the sophistication required to model with good enough fidelity the flow through the main rotor of a helicopter can be achieved by representing the main rotor as a thin 3-dimensional surface of micro-jets (current controlled current sources) which morphs into various shapes and velocity distributions as a function of flight conditions and flight control inputs. Operation of each micro-jet can be determined by establishing a force balance between the force generated by massflow through the micro-jet and the usual aerodynamic lift and drag forces created by a wing.

3.4 Flexible Disk Predictions (Develop testable predictions)

Based on the Floppy Disk Hypothesis we now assert that the sophistication of the **Main Rotor Model** can be expanded in steps until a point is reached where the accuracy of the model is good enough for the purposes of the PRLS. For the moment this judgement will be based on subjective considerations until finally verified with the overall **PRLS model** (we will return to this point in Chapter 8).

3.5 The Main Rotor Model (Gather data to test predictions)

The **Main Rotor Model** will now be developed based on observational evidence of paragraph 3.1 and the Flexible Disk Hypothesis. To do this the following assumptions are made:

- The blades are rigid, flapped and capable of adjusting in pitch as function of swash plate position.
- The radial orientation of the blades are dominated by an equilibrium between centrifugal force and lift, thus sweeping out a variable (floppy) surface referred to as the Main Rotor Cone.
- The Main Rotor Cone is sub-divided into a fine mesh with each sub-division (control surface) behaving like a micro-jet (orifice) that obeys the Rocket Equation.
- The blades generate the ordinary aerodynamic forces of lift and drag as they sweep the Main Rotor Cone.
- With the blade underneath a given orifice the lift and drag of the Rotor Cone in that location is generated by the inflow into the orifice being accelerated by the blade such as to produce the combined force as observed.

3.5.1 Modelling the Main Rotor as a flat actuator disk

As first improvement a variation of the blade element theory originally formulated by *Froude* [53] and as applied by *Chollet* [54] for a helicopter in hover can be considered. Note that the free literature abound with various papers on the blade element theory with criticism from some researchers such as *Van Kuik* [56] who points out that the theory under-predicts the inflow velocity with 10% to 15% compared with experimental evidence. His solution is to modify the

theory by adding “edge forces” acting orthogonal to the flow, thus performing no work which can upset the conservation of energy rule as demanded by blade element theory.

On the other hand, supportive evidence for blade element theory is also forthcoming from some researchers such as *Mahmuddin* [55] who extended the model by including the effects of tip losses as proposed by *Prandtl*, achieving good results in a windmill application. It would appear that blade element theory is an attempt to represent a very complicated process with a relatively simple model. Some deviation from reality should therefore be expected when using this model.

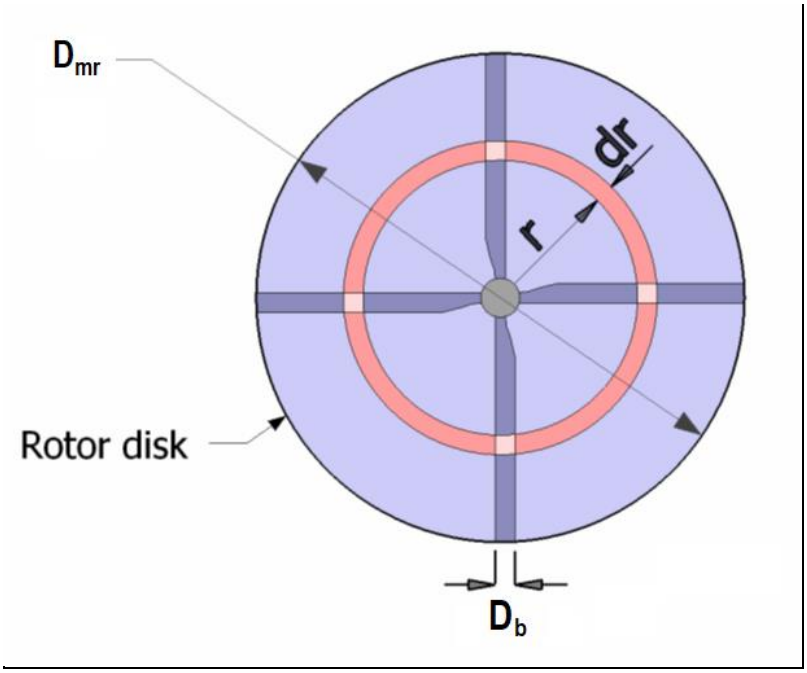


Figure 80. Flat four-bladed rotor disk with annular control surface [A Landman, 2016]

Consider a flat and rigid actuator disk comprising a rotor and four symmetric blades at a given pitch and spinning at constant angular rate ω_{mr} with annular control surface of radius *r* and width *dr* as illustrated in Figure 80. In such a configuration the lift generated by the control surface must be the lift generated by the four overlap sections between rotor blades and annular control surface as shown. Applying Equation 24, the lift ΔFL_{mr} generated by the four sections is given by:

$$\Delta FL_{mr} = 4\left[\frac{\rho_{mr}\left(r\omega_{mr}\right)^2 CL_{\alpha mr}\alpha_{mr}D_b\,dr}{2}\right] \qquad \textbf{Equation 42}$$

With $CL_{\alpha mr}$ the slope of the lift coefficient curve of the rotor blade, α_{mr} the angle of attack between rotor blade cord and air entering the rotor disk and D_b the width of the rotor blade. Note that α_{mr} is a function of both the rotational speed of the rotor and the speed of the air entering the rotor disk. Based on Thin Airfoil Theory as formulated by *Munk* [47] the value of $CL_{\alpha mr}$ should be 2π if the rotor blades are infinite in length (i.e. if there are no vortex flows at the tips of the rotor blades). For an Aspect Ratio of 12 as is the case of a Rooivalk Main Rotor Blade this is probably a reasonable assumption.

Deliberation of Equation 42 reveals one of the basic problems confronted by this study – the force exerted on the air over a small section of the actuator disk is dependant on the angle of attack at that location but the angle of attack is dependant on the velocity of air at the same location. The velocity of the air at that location is in turn dependant on how the entire air mass

around the helicopter reacts to each and every one of the forces acting on all sections of air over the entire actuator disk. The latter statement follows from the fact that the air around the helicopter reacts in similar fashion to a 3-dimensional transmission line (with the Maxwell Equation imposing a similar response as the Navier Stokes Equation). Any disturbance at any location in this network will result in different responses throughout the network at different times later. Complicating the problem even more is the fact that things such as the tail rotor, fuselage, engine outflows and ground also affect this flow pattern. To solve this problem *Bludau*, *Rauleder*, *Friedmann* and *Hajek* [57] devised a training simulator in which a main rotor model was coupled with a fluid dynamics model that considers ground objects. This was necessary to emulate the changes in flight characteristics that occur when a helicopter is flown close to objects such as ships and buildings.

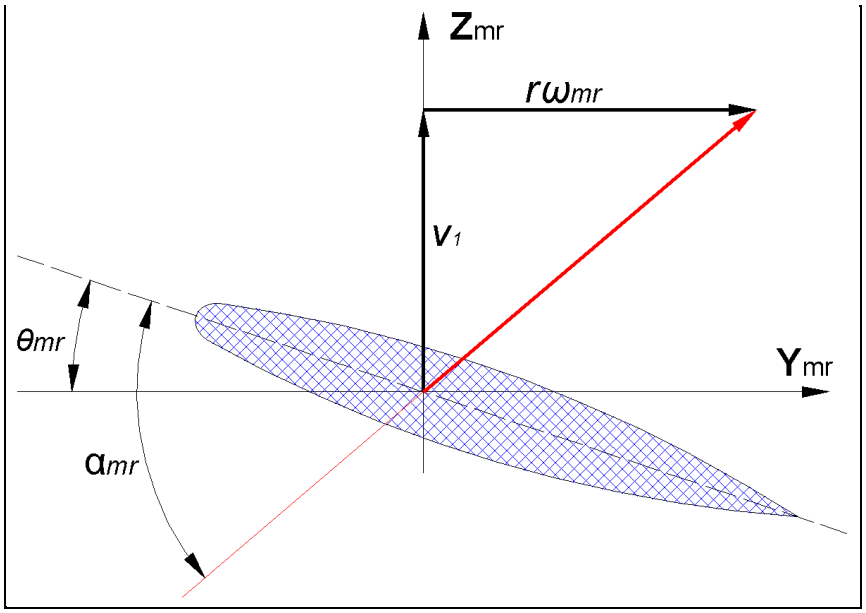


Figure 81. Angle of attack at control surface [A Landman, 2016]

For the moment, assume that the inflow is homogeneous, travelling at speed $v1$ normal to the rotor disk surface. Note that this is equivalent to assuming that each segment in Figure 80 acts like a constant current source (i.e. air density is constant while all inflow velocities are forced to be the same irrespective of massflow rate). Under this assumption the angle of attack α_{mr} is given by:

$$\alpha_{mr} = atan\left(\frac{v_1}{r\omega_{mr}}\right) - \theta_{mr} \qquad \textbf{Equation 43}$$

Assuming $CL_{\alpha mr}$ to be constant (good approximation for symmetric profile used for blades of Rooivalk AH-2A Main Rotor) the overall lift generated by the Main Rotor of Rooivalk AH-2A can now be calculated by substituting Equation 43 into Equation 42 and integrating over the entire rotor disk surface except the Rotor Head (about 10% of the inner rotor disk area) so that:

$$FL_{mr} = 2\rho_{mr}\omega_{mr}^2 CL_{\alpha mr} D_b \int_{0.05D_{mr}}^{0.5D_{mr}} \left[\text{atan}\left(\frac{v_1}{r\omega_{mr}}\right) - \theta_{mr}\right] r^2 dr \qquad \textbf{Equation 44}$$

In the case of Rooivalk D_b equals 0,6 meter and ω_{mr} equals 27,489 rad/sec and ρ_{mr} exuals 1,2929 kg/m^3 at 0°C and 1 ATM. Assuming $CL_{\alpha mr}$ to have a value of 2π and noting that the

Main Rotor Blade used on Rooivalk AH-2A is symmetrical in profile the lift generated by the main rotor disk of Rooivalk AH-2A was calculated as function of inflow wind speed v_1 and pitch angle θ_{mr} under these conditions. The result is illustrated in Figure 82 from which it is evident that with the aircraft in hover a blade pitch angle of 6° is required to produce a lift force of 80000 Newton at an inflow speed of 5 m/sec.

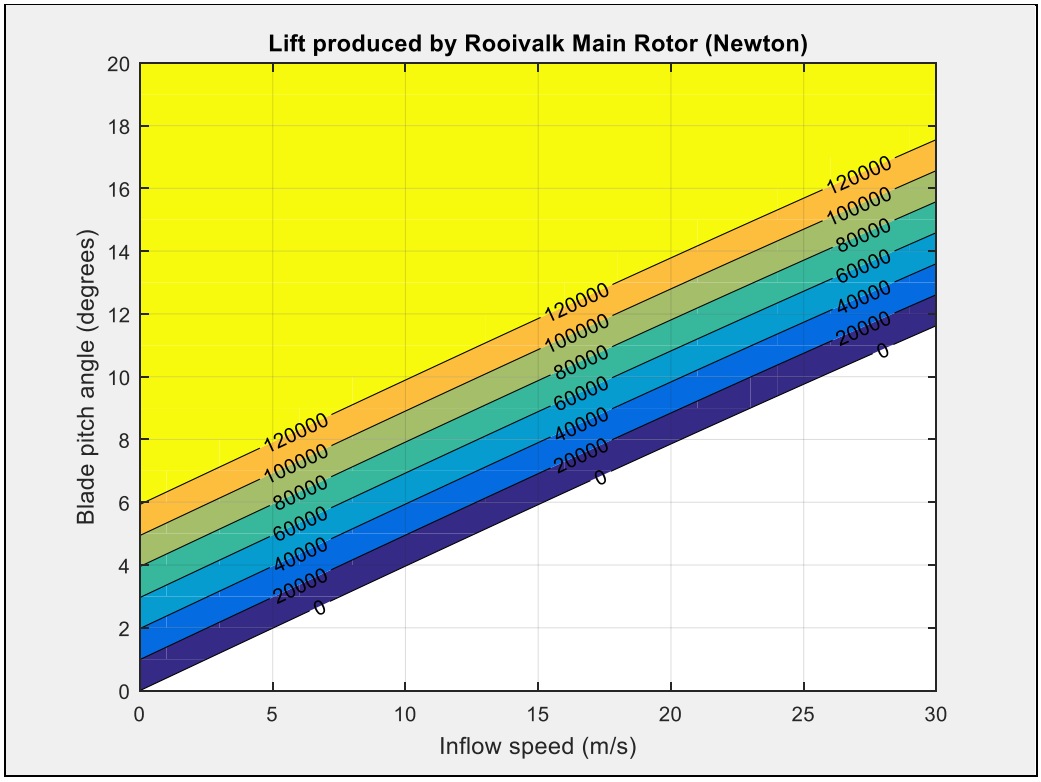


Figure 82. Estimated Main Rotor lift: flat actuator disk [A Landman, 2018]

Equation 44 presents a number of problems from the PRLS perspective. First, it assumes the inflow wind velocity to be homogenous and constant over the entire actuator disk while it varies greatly in reality. Second, it represents a one-dimensional model of the downwash which is only applicable for a helicopter in the hover - changes in downwash due to Cyclic Lever control inputs and ambient wind are not considered. It is evident that such a model would yield an estimate of actuator disk outflow with such large errors that it would not be suitable for the PRLS system.

3.5.2 Introducing the effects of Cyclic Control

In the previous paragraph, the assumption was that the downwash is radially symmetric (i.e. one-dimensional) and pointed along the Main Rotor Drive Axis. This assumption is only applicable during steady state flight conditions such as hovering with no ambient wind. Considering the flight envelope and environmental conditions under which rockets are to be launched from Rooivalk AH-2A, such conditions would only occur on rare occasions. It therefore has to be assumed that some Cyclic Lever control inputs will be present, in which case the downwash will have an asymmetric velocity distribution. This point is important because as shown in Chapter 2 the FZ90 Rocket is very sensitive to rotor downwash. This sensitivity is so high that the changes in downwash speed and direction caused by things such as Collective Lever and Cyclic Lever control inputs are bound to cause variations in the FZ90 Rocket trajectory that are beyond the accuracy limits as postulated by the PRLS hypothesis of paragraph 1.3.

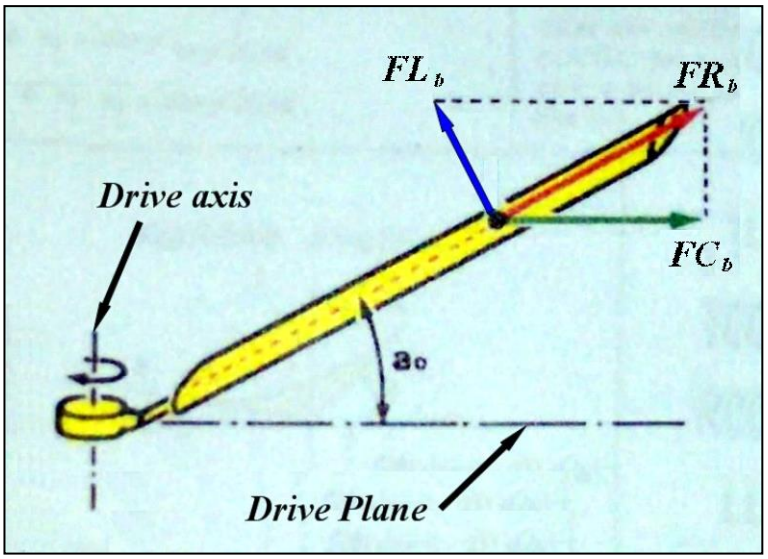


Figure 83. Force equilibrium with rotor blade at coning angle a_o [Raletz, 1988]

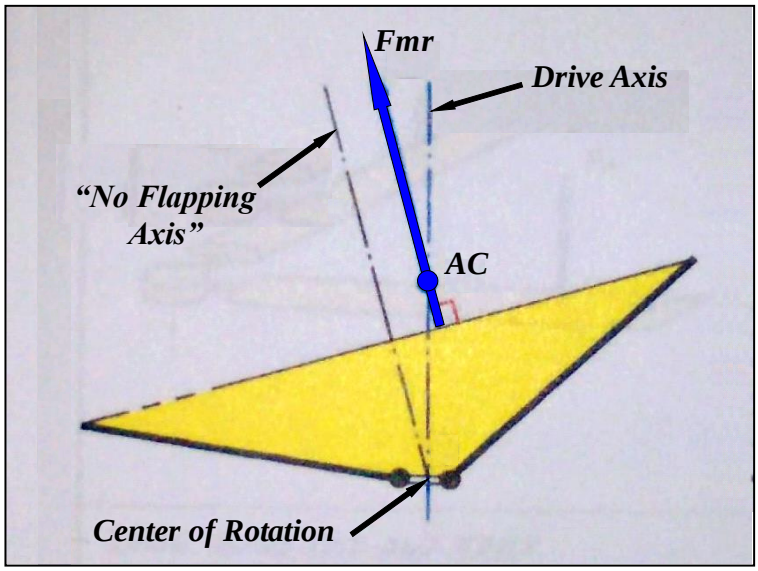


Figure 84. Force generated by Main Rotor with rotor cone tilted [Raletz, 1988]

To develop a relationship for the velocity distribution over the Rotor Cone as function of the positions of the Collective Lever and Cyclic Lever, note that each blade of the Main Rotor of Rooivalk AH-2A is fitted with a flapping hinge. As described by *Raletz* [58], the flight paths of the blades of such a rotor form a cone around the so-called "No Flapping Axis" as illustrated in Figure 84. Since each blade is hinged, they follow flight paths that can be described with a so-called coning angle a_o as illustrated in Figure 83. As shown, the coning angle represents a condition of force equilibrium where the vector sum of blade weight $\boldsymbol{FW}_b$ and centrifical force $\boldsymbol{FC}_b$ acting through the CoM of the blade and lift $\boldsymbol{FL}_b$ generated by the blade lays along the length of the blade. Since the weight of the blade is small compared with the lift and centifical forces of the blade and the rotation rate of the rotor is constant, the coning angle is primarily determined by $\boldsymbol{FC}_b$ and $\boldsymbol{FL}_b$ (which is adjusted by the Pilot to generate the lift required from the Main Rotor to lift and maneuver the Aircraft).

During hover with no ambient wind the No Flapping Axis co-incides with the Drive Axis. Under these conditions the Rotor Cone is symmetric around the Drive Axis, resulting in a constant coning angle of about 2,7° for Rooivalk AH-2A with all-up mass at 8000 kg. For all other flight conditions the coning angle of each blade changes cyclically for each rotation of the Main Rotor.

It is evident that as flight conditions change the inflow will exhibit a variable two-dimensional velocity distribution over the Rotor Cone.

The effect of a Collective Lever control input is for the Swash Plate to raise, causing the lift generated by all the blades to increase with the same amount. Conversely, the effect of a Cyclic Lever control input is for the Swash Plate to tilt, causing the lift generated by the blades to increase in one half of the rotor disk and decrease in the other half. The result is a change in the flight paths followed by the rotor blades, causing the Rotor Cone to tilt in the same direction as the Cyclic Lever displacement. The end effect is a torque imbalance as illustrated in Figure 84 that induces a change in roll or pitch of the aircraft. Since the aircraft is a helicopter, it behaves as if standing on top of a sphere – any change in roll or pitch causes it to slide off towards the ground. To maintain the same height under these conditions the Pilot must increase the Collective Lever setting, causing the aircraft to not only maintain height but also move in the direction of the disturbance.

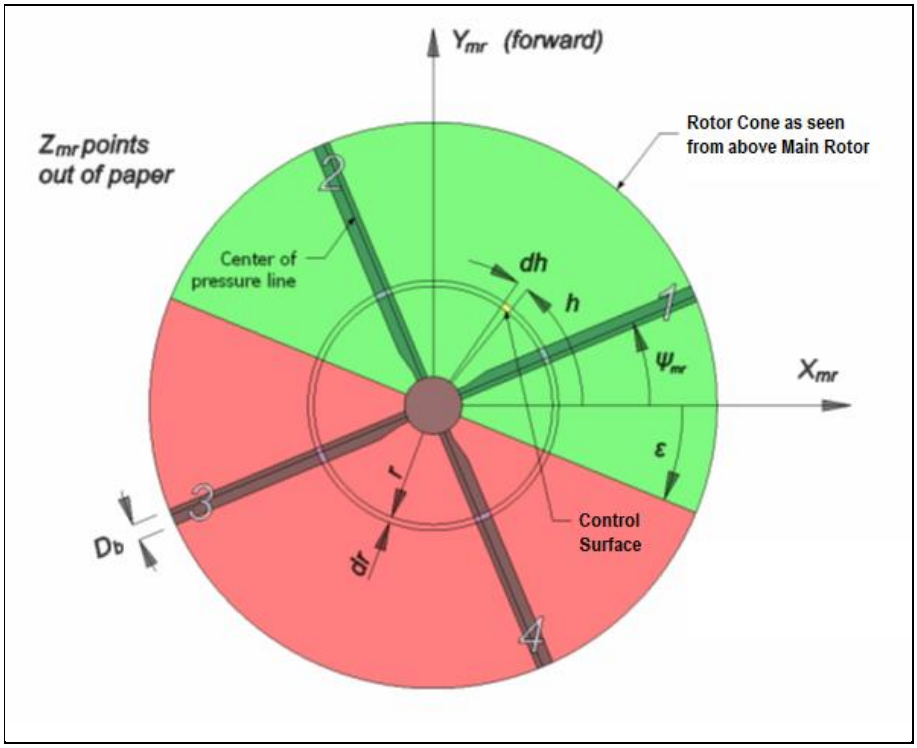


Figure 85. Test surface on Rotor Cone with flapping blades [A Landman, 2016]

It should therefore be possible to calculate the velocity distribution of inflow over the Main Rotor Cone provided the positions of the Collective Lever and Cyclic Lever are known (measured). To this effect, a right-handed coordinate system known as Main Rotor Axes[7] is now defined with the $\boldsymbol{Z}_{mr}$ axis aligned with the drive axis and the $\boldsymbol{Y}_{mr}$ axis pointing towards the front of the aircraft as illustrated in Figure 85. To calculate the inflow a control surface is introduced at arbitrary position with position and dimensions as shown. Note that the control surface is stationary and does not move with the blades.

The effect of Cyclic Lever position is illustrated in Figure 85 where the pink area represents positions where the blade pitch angle is increased and the green area positions where the blade pitch angle is decreased (i.e. the Collective Lever is pushed towards the forward right in the example shown). The transition between the two lifting zones is demarcated by a transition line through the center of the Main Rotor at Cyclic Steer Angle ε as shown. Angle ε is such that the transition line is normal to the offset direction of the Cyclic Lever from it's neutral position with the green segment located in the direction of the Cyclic Lever offset. Note that anti-clockwise

[7] An axes system pitched forward at 5° with reference to Aircraft Axes on Rooivalk AH-2A.

angles are positive since the $\boldsymbol{Z}_{mr}$ axis points out of the paper. Hence, in the example of Figure 85 the Collective Lever is roughly at a steer angle of -25°.

As the Main Rotor rotates the pitch angle θ_{mr} of each rotor blade is modulated sinusoidally[8] by the Swash Plate as function of the Main Rotor Azimuth Angle ψ_{mr}. Due to the mechanics of the Swash Plate, each Main Rotor Blade follows exactly the same modulation pattern but shifted in rotor azimuth with even multiples of 90°. Taking Main Rotor Axes as reference frame and assuming that Bade1 is initially aligned with the $\boldsymbol{X}_{mr}$ axis the modulation process for Blade1 can now be described with the following relationship:

$$\theta_{mr} = BPOA + BPMA * \sin\left(\psi_{mr} + \varepsilon\right) \qquad \textbf{Equation 45}$$

with θ_{mr} the blade pitch angle of Blade1, *BPOA* the blade pitch offset angle as determined by the Collective Control lever, *BPMA* the blade pitch modulation angle as determined by the Cyclic Control Lever, ψ_{mr} the Main Rotor Azimuth Angle and ε the Cyclic Control Lever steer angle. Note that Equation 45 assumes each blade to rotate in accordance with a ZY'X" rotation scheme. Hence, anti-clockwise rotations in Figure 85 are positive because the $\boldsymbol{Z}_{mr}$ axis points out of the paper.

At the control surface the angle of attack α_{mr} is a function of θ_{mr} and the speed v_1 at which the air enters the rotor disk at the control surface as well as the orbital velocity of the blade at the control surface. The situation is illustrated in Figure 81 for both v_1 and θ_{mr} negative, yielding the relationship as formulated in Equation 43.

Note: *Equation 43 and Equation 45 can be confusing unless some rules are strictly enforced. For the purpose of the analysis here all angles and vectors are measured in terms of Main Rotor Axes and assuming a ZY'X" rotation scheme. In the example of Figure 81 the orbital velocity of the blade is always as shown because the Main Rotor allways rotates in anti-clockwize direction. In the example the inflow is parallel to the $\boldsymbol{Z}_{mr}$ axis so that v_1 is a positive entity and the left hand term in Equation 43 yields a positive value. The blade pitch angle θ_{mr} as shown is a negative entity because it represents a negative rotation around the X''_{mr} axis using the right-hand rule. Substituted in Equation 43 it results in an increase in angle of attack. Finally, the angle of attack α_{mr} is a positive entity because the blade must be rotated through a positive angle around the X''_{mr} axis for the blade to face into the incident air flow.*

The distribution of angle of attack over the Main Rotor Disk as predicted by Equation 43 and Equation 45 was calculated assuming a value of -4° for BPOA, +2° for BPMA, +45° for α and -0,5 m/s for v_1. The result is illustrated in Figure 86.

With α_{mr} at hand the wind speed directly below the control surface can now be derived by employing the force equivalence principle as formulated in paragraph 3.2. To do this it is noted that each blade generates a temporary force when it passes over the control surface and that the forces generated by all of the blades must add in some or other way. The resultant force must also result in some or other airflow through the control surface. In other words the control surface becomes a small square orifice instead of the annulus as assumed by *Froude* [53] and *Chollet* [54] or the toroidal jet as assumed in paragraph 2.5.8.4. Also note that each blade is only present over the control surface for a fraction of the time during which the Main Rotor completes one revolution and that this fraction decreases for control surfaces located towards the outside of the Main Rotor Disk.

[8] This approximation is reasonable since the maximum Blade Pitch Angle is only a few degrees.

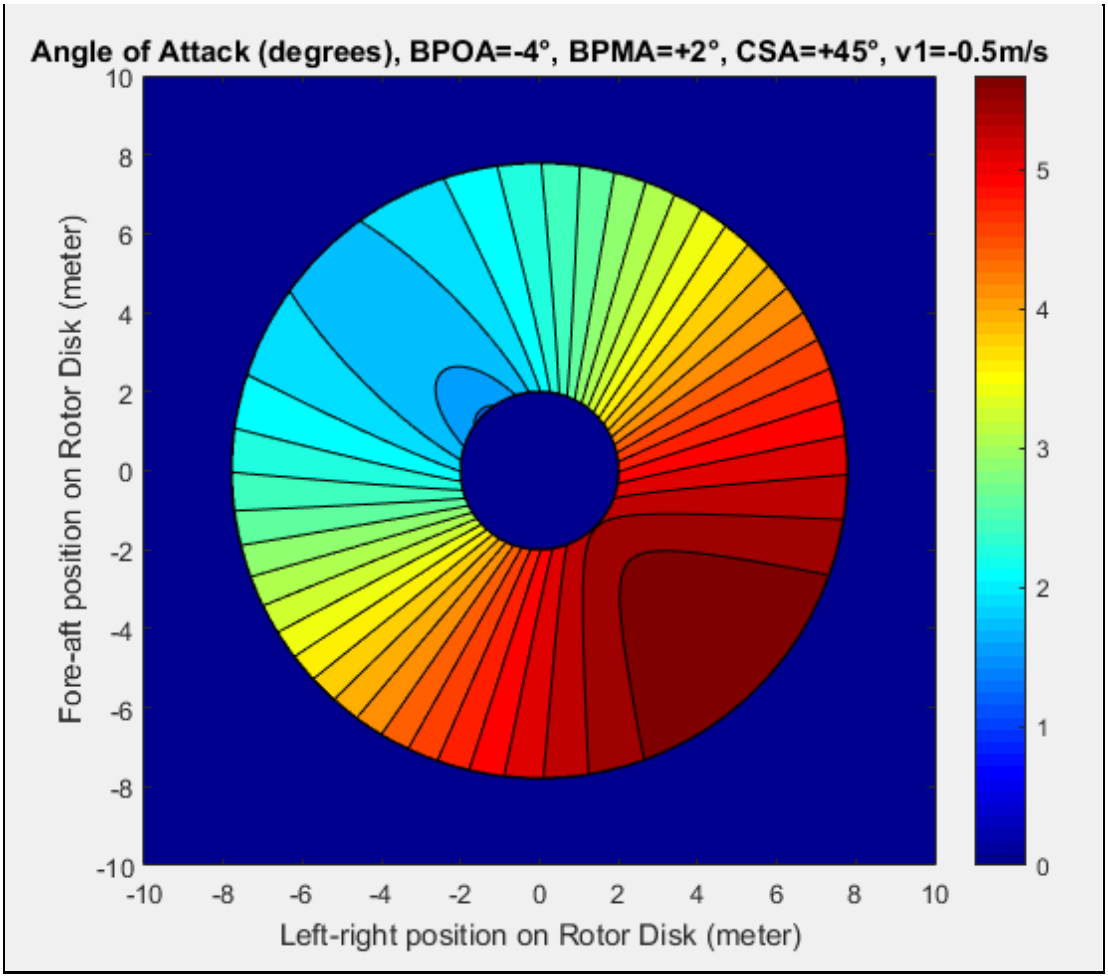


Figure 86. Typical distribution of Angle of Attack over Rotor Disk [A Landman, 2016]

How can all of this be reconciled mathematically? To answer this question, refer to Figure 85 in which the lift generated by the blade segment is assumed to work through the centre of pressure of the blade. Overall, the lift generated by a blade can therefore be seen as a Lift Line along the length of the blade that sweeps the surface of the Main Rotor Cone. Hence, for a given *Δr* the force that would act on the control surface as a blade sweeps over it can be calculated with Equation 24. Figure 87 shows a typical example of the forces that would be exerted on an arbitrarily chosen control surface under these conditions. It is evident that the force acting on the control surface is characterized by a duty cycle β_{cs} as function of dh only:

$$\beta_{cs} = \frac{2dh}{\pi} \qquad \textbf{Equation 46}$$

As *Δh* is made smaller, the force exerted on the control surface becomes shorter in duration until it can be approximated as a square wave (impulse) with frequency equal to four times the rotational frequency of the Main Rotor and amplitude equal to the lift force generated by the blade segment at center of the control surface and duty cycle determined by the value of *Δh.* Hence, in the limit when both *Δh* and *Δr* approaches zero it should be possible to derive a smooth surface for the outflow distribution.

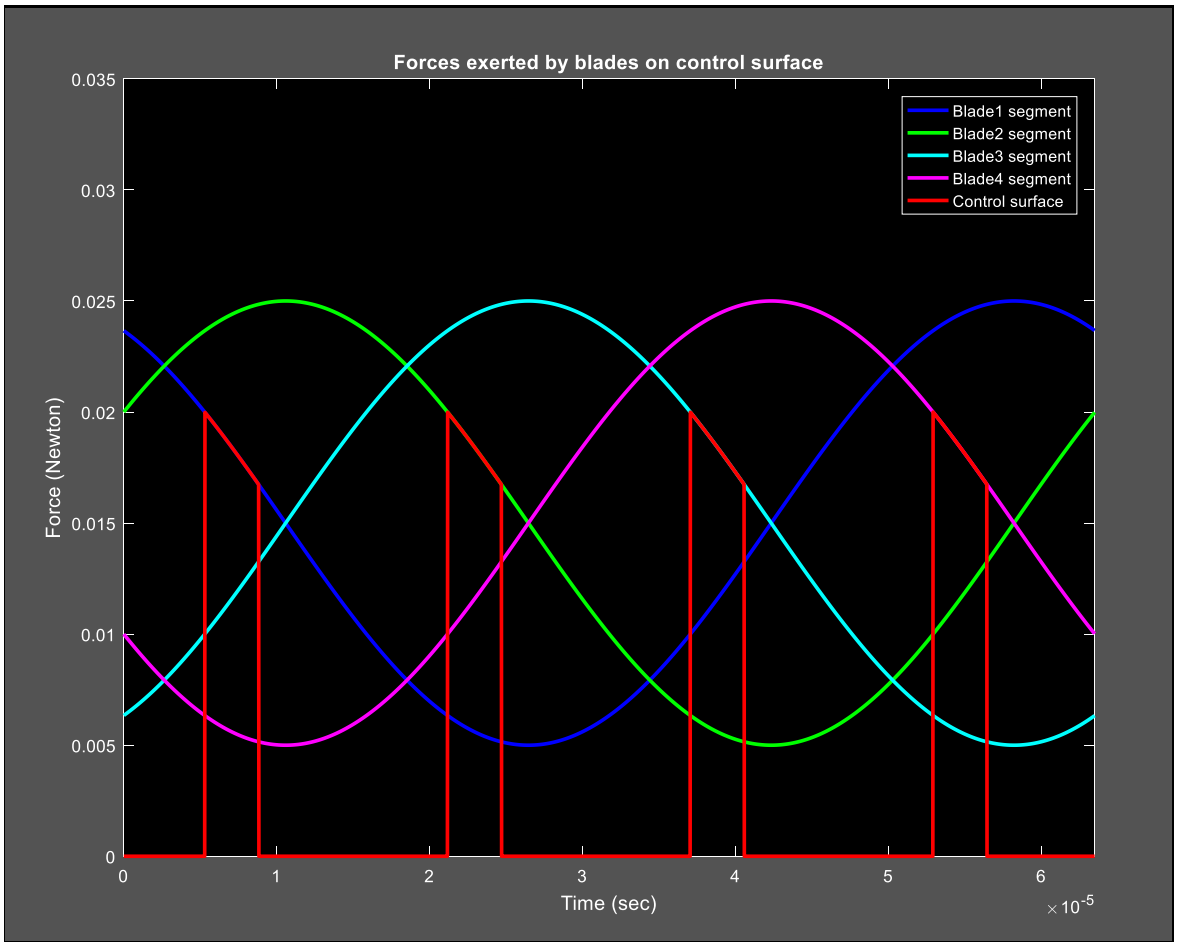


Figure 87. Forces exerted by four blades on typical control surface [A Landman, 2016]

Setting Equation 35 and Equation 42 equal and introducing the duty cycle as determined by dh for the force curve as in Figure 87 the force equilibrium at the control surface is given by:

$$\left[\frac{\rho_{mr}\left(r\omega_{mr}\right)^2 CL_{amr}\alpha_{mr}D_b dr}{2}\right]\left(\frac{2dh}{\pi}\right)=\rho_{mr}\, r\, dh\, dr\, v_1\left(v_2-v_1\right) \qquad \textbf{Equation 47}$$

$$\Rightarrow \quad v_2=\frac{2r\omega_{mr}^2 CL_{amr}\alpha_{mr}D_b}{\pi v_1}+v_1$$

Substituting Equation 43 and Equation 45 into Equation 47 and noting that with Blade1 over the test surface $\psi_{mr}=h$ the outflow velocity v_2 just below the control volume is given by:

$$v_2=\frac{2r\omega_{mr}^2 CL_{amr}\left[\operatorname{atan}\left(\frac{v_1}{r\omega_{mr}}\right)-\left\{BPOA+BPMA\,\sin\left(h-\varepsilon\right)\right\}\right]D_b}{\pi v_1}+v_1 \qquad \textbf{Equation 48}$$

Where v_2 is now a function of both *r* and *h* so that it represents a two-dimensional velocity distribution of the outflow. The outflow of Rooivalk AH-2A was calculated with Equation 48 assuming values of -4° for BPOA, +2° for BPMA, +45° for α, -0,5 m/s for v_1 and 2π for $CL_{\alpha mr}$ as

a typical scenario directly after application of the Cyclic Lever control input. The result is illustrated in Figure 88.

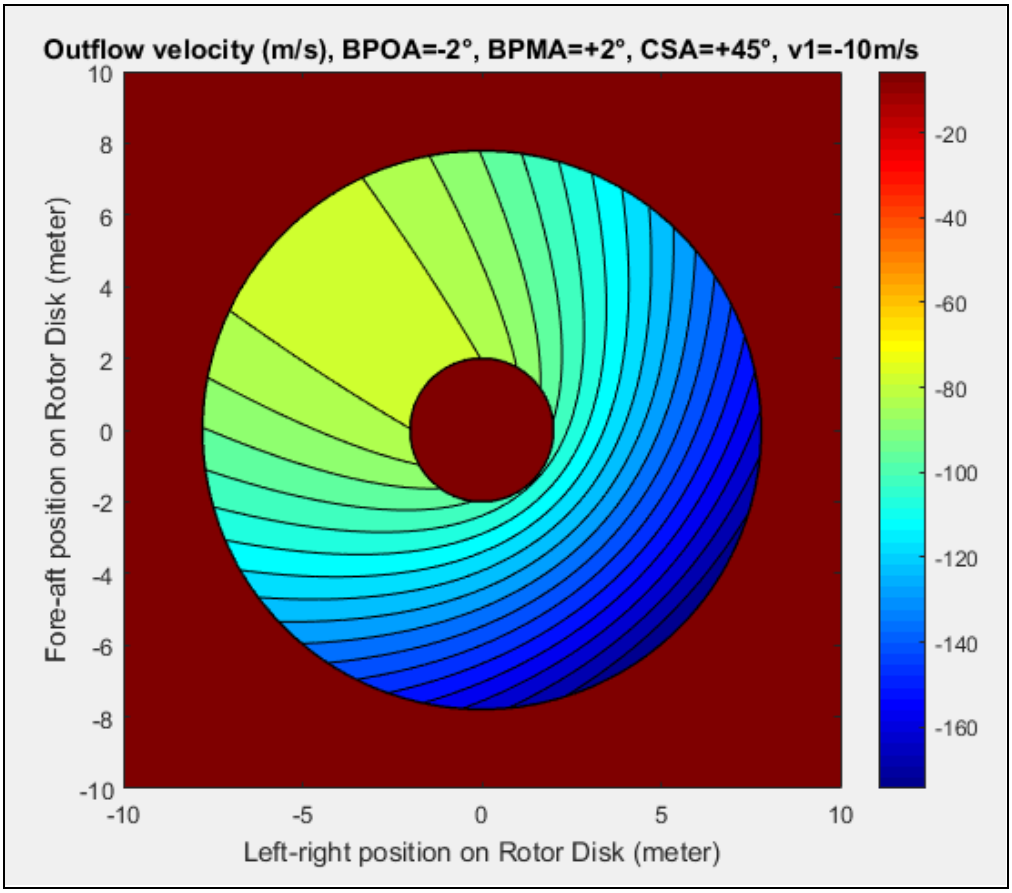


Figure 88. Example of outwash velocity directly after Cyclic Lever input [A Landman, 2016]

It must be noted here that Figure 88 represents a momentary state with the aircraft stationary and no ambient wind. It also assumes a homogenous inflow of -10 m/s which will not occur in reality. Instead, the inflow is not constant nor homogenous, resulting in an outflow velocity distribution that constantly changes with flow pattern more complex than Figure 88. Application of a Cyclic Lever control input as depicted in Figure 88 will cause the aircraft to accelerate and move through the ambient air mass, causing lateral speed components of the air above the rotor disk that will introduce assymmetric components in the downwash. It is evident that the rotor downwash of a helicopter is a dynamic entity, presenting continually changing and different wind conditions to any rocket launched from any of the weapon stations.

3.5.3 Introducing Lateral Dissymmetry

In the case of ambient wind or with the aircraft moving, the inflow will have lateral velocity components in the plane of the Rotor Disk. These lateral components will add to the orbital velocities of advancing and retreating rotor blades, causing changes in lift and angle of attack of the rotor blades that will vary with position over the Rotor Disk.

To derive a relationship for these effects, consider a Main Rotor comprising a rotor head with four rigid blades attached through flapping hinges as illustrated in Figure 89. Located on Blade1 is a blade segment with radial width dr_b at radial distance r_b from the flapping hinge centre. Incident on the blade segment is air that moves with velocity $\boldsymbol{VI}$ which is the vector sum of the orbital velocity $\boldsymbol{V}_{bs}$ of the blade segment, local wind velocity $\boldsymbol{WS}$ and aircraft velocity $\boldsymbol{AS}$. Note

that in this arrangement, all velocities and angles are measured using the right hand rule with respect to Main Rotor Axes as illustrated in Figure 89. The incident air approaches the blade segment at some or other angle of attack, creating lift orthogonal to vector $\boldsymbol{VI}$ and the length of the blade. The total lift generated by the blade is therefore the sum of all the lift forces generated by all the segments over the entire blade. At the same time, the blade is subjected to a centrifical force so that it settles into an attitude with coning angle a_o where the force equilibrium as illustrated in Figure 83 is established.

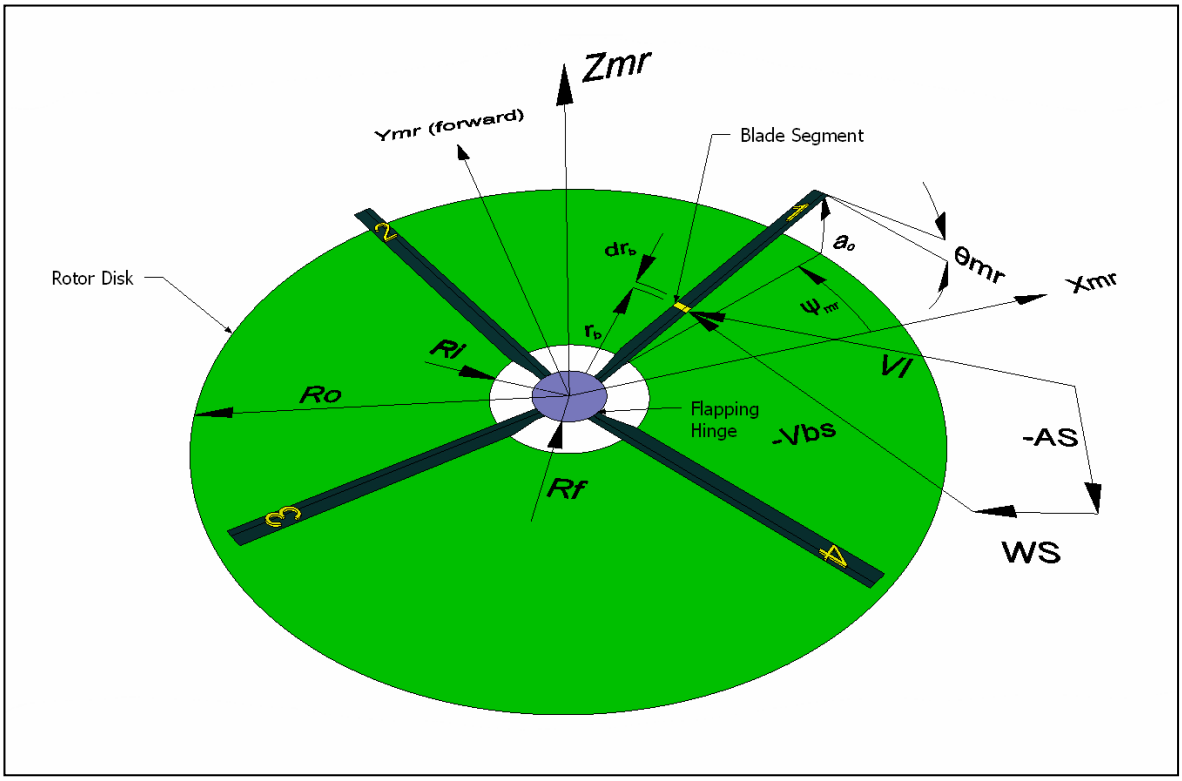


Figure 89. Arrangement to calculate effect of ambient wind and aircraft movement [A Landman, 2016]

To calculate the angle of attack of air incident on the blade segment shown in Figure 89, introduce unit vector $\hat{b}$ that points from the root to the tip of Blade1. Applying the ZY'X" direction cosine matrix of Appendix A and noting that in Blade Coordinates $\hat{b} = [1,0,0]$ yields the following equation for $\hat{b}$ expressed in Main Rotor Axes:

$$\hat{b} = \cos(\psi_{mr})\cos(a_o)\hat{x}_{mr} + \sin(\psi_{mr})\cos(a_o)\hat{y}_{mr} - \sin(a_o)\hat{z}_{mr} \qquad \textbf{Equation 49}$$

With $\hat{x}_{mr}$, $\hat{y}_{mr}$ and $\hat{z}_{mr}$ unit vectors along the $\boldsymbol{X}_{mr}$, $\boldsymbol{Y}_{mr}$ and $\boldsymbol{Z}_{mr}$ axes of Main Rotor Axes and with the sign of the angles determined using the right-hand rule (i.e. angle ψ_{mr} is a positive entity while angles a_o and θ_{mr} are negative entities at the particular moment as illustrated in Figure 89). Applying the ZY'X" direction cosine matrix of Appendix A and noting that the orbital velocity of the blade segment is always in the rotor disk plane yields the following equation for the orbital velocity $\boldsymbol{V}_{bs}$ of the blade segment in Main Rotor Axes:

$$\mathbf{V}_{bs} = -\omega_{mr}\left[R_f + r_b\cos(a_o)\right]\sin(\psi_{mr})\,\hat{x}_{mr} + \omega_{mr}\left[R_f + r_b\cos(a_o)\right]\cos(\psi_{mr})\,\hat{y}_{mr} + 0\,\hat{z}_{mr} \qquad \textbf{Equation 50}$$

Note that w_{mr} is positive for the rotor rotating in an anti-clockwise fashion (i.e. along the $\boldsymbol{Z}_{mr}$ axis) so that it is a negative entity in the case of Rooivalk AH-2A. The Inflow velocity $\boldsymbol{V}_{in}$ is now defined as follows:

$$\mathbf{V}_{in} = \mathbf{WS} - \mathbf{AS} \qquad \textbf{Equation 51}$$

With ambient wind velocity $\boldsymbol{WS}$ and aircraft velocity $\boldsymbol{AS}$ as defined before. Since the velocity of stationary air incident on the blade segment is antiparallel to the orbital velocity of the blade segment, the velocity $\boldsymbol{VI}$ of air incident on the blade segment from the perspective of the blade segment is given by:

$$\begin{aligned}\mathbf{VI} \;=\; & \left[\omega_{mr}\left[R_f + r_b\cos\left(a_o\right)\right]\sin\left(\psi_{mr}\right) + WS_x - AS_x\right]\hat{x}_{mr} + \\ & \left[-\omega_{mr}\left[R_f + r_b\cos\left(a_o\right)\right]\cos\left(\psi_{mr}\right) + WS_y - AS_y\right]\hat{y}_{mr} + \\ & \left[WS_z - AS_z\right]\hat{z}_{mr}\end{aligned} \qquad \textbf{Equation 52}$$

with WS_x , WS_y and WS_z the components of ambient wind velocity in Main Rotor Axes. Similarly AS_x , AS_y and AS_z are the components of aircraft velocity along the $\boldsymbol{X}_{mr}$, $\boldsymbol{Y}_{mr}$ and $\boldsymbol{Z}_{mr}$ axes of Main Rotor Axes. Next, define unit vector $\hat{c}$ which points from the leading edge to the trailing edge of the blade segment. Applying the ZY'X" direction cosine matrix of Appendix A yields:

$$\begin{aligned}\hat{c} \;=\; & \left[-\sin\left(\psi_{mr}\right)\cos\left(\theta_{mr}\right) + \cos\left(\psi_{mr}\right)\sin\left(a_o\right)\sin\left(\theta_{mr}\right)\right]\hat{x}_{mr} + \\ & \left[+\cos\left(\psi_{mr}\right)\cos\left(\theta_{mr}\right) + \sin\left(\psi_{mr}\right)\sin\left(a_o\right)\sin\left(\theta_{mr}\right)\right]\hat{y}_{mr} + \\ & \left[+\cos\left(a_o\right)\sin\left(\theta_{mr}\right)\right]\hat{z}_{mr}\end{aligned} \qquad \textbf{Equation 53}$$

In terms of lift generation, the velocity of air incident on the blade segment comprises the vector sum of two components. The first component $\boldsymbol{VP}$ is parallel to unit vector $\hat{b}$ and generates neither lift nor rotary drag. It is given by the following relationship:

$$\mathbf{VP} = \left(\hat{b}\cdot\mathbf{VI}\right)\hat{b} \qquad \textbf{Equation 54}$$

The second component $\boldsymbol{VT}$ is tangential to unit vector $\hat{b}$ and is the component that actually generates both lift and rotary drag. It is given by the following relationship:

$$\mathbf{VT} = \mathbf{VI} - \mathbf{VP} \qquad \textbf{Equation 55}$$

Suitable manipulation of the dot-product between vector $\boldsymbol{VT}$ and unit vector $\hat{c}$ should allow determination of the angle of attack α_{bs} of air incident on the blade segment. To do this define:

$$VT = \sqrt{VT_x^2 + VT_y^2 + VT_z^2} \qquad \textbf{Equation 56}$$

So that:

$$\hat{e} = \frac{\mathbf{VT}}{VT}$$ **Equation 57**

with VT_x , VT_y and VT_z the components of vector $\boldsymbol{VT}$ along the the $\boldsymbol{X}_{mr}$, $\boldsymbol{Y}_{mr}$ and $\boldsymbol{Z}_{mr}$ axes of Main Rotor Axes and $\hat{e}$ a unit vector that lies parallel to vector $\boldsymbol{VT}$. Then follows:

$$\left|\alpha_{bs}\right| = \operatorname{acos}\left(\hat{c}\cdot\hat{e}\right)$$ **Equation 58**

To determine the sign of α_{bs} note from Figure 89 that the cross-product between unit vectors $\hat{c}$ and $\hat{e}$ is parallel to unit vector $\hat{b}$ when α_{bs} is positive, and antiparallel when α_{bs} is negative. This yields the following relationship:

$$\alpha_{bs} = \left|\alpha_{bs}\right| \;\; if \;\; \hat{b}\cdot\left(\hat{c}\times\hat{e}\right) \geq 0$$
$$else \;\; \alpha_{bs} = -\left|\alpha_{bs}\right|$$ **Equation 59**

To determine the lift force $d\boldsymbol{FL}_{bs}$ generated by the blade segment define unit vector $\hat{g}$ as follows:

$$\hat{g} = \left(\hat{b}\times\hat{e}\right)$$ **Equation 60**

So that the lift force $d\boldsymbol{FL}_{bs}$ is given by:

$$d\mathbf{FL}_{bs} = \left(\frac{\rho_{mr} VT^2 CL_{\alpha mr}\alpha_{bs} D_b dr_b}{2}\right)\hat{g}$$ **Equation 61**

Note that both the magnitude and direction of the lift force $d\boldsymbol{FL}_{bs}$ changes with position along the length of the blade. The total lift $\boldsymbol{FL}_b$ generated by Blade1 is the vector sum of all the lift forces generated by all the blade segments over the entire blade:

$$\mathbf{FL}_b = \int_{R_i}^{R_o} \frac{\rho_{mr} VT^2 CL_{\alpha mr}\alpha_{bs} W_{mr}}{2}\,\hat{g}\; dr_b$$ **Equation 62**

with the limits R_i and R_o as illustrated in Figure 89. Note that the integral of Equation 62 actually represents three distinct integrals along the three axes of Main Rotor Axes. To derive an equation for the centrifugal force $\boldsymbol{FC}_b$ acting on Blade1 some inspection of Figure 89 and applying the ZY'X" direction cosine matrix of Appendix A yields the following:

$$\mathbf{FC}_b = m_b\omega_{mr}^2\left[R_f + R_{com}\cos\left(a_o\right)\right]\left[\cos\left(\psi_{mr}\right)\hat{x}_{mr} + \sin\left(\psi_{mr}\right)\hat{y}_{mr} + 0\hat{z}_{mr}\right]$$ **Equation 63**

with m_b the mass of the rotor blade and R_{com} the distance from the flapping hinge center to the Center of Mass of Blade1. Hence, the resultant force $\boldsymbol{FR}_b$ acting on Blade1 is given by:

$$\mathbf{FR}_b = \mathbf{FL}_b + \mathbf{FC}_b \qquad \textbf{Equation 64}$$

Next, define a unit vector $\hat{a}$ parallel with vector $\boldsymbol{FR}_b$ so that the following must hold:

$$\hat{a} = \frac{\mathbf{FR}_b}{\sqrt{FR_{bx}^2 + FR_{by}^2 + FR_{bz}^2}} \qquad \textbf{Equation 65}$$

with FR_x , FR_y and FR_z the components of vector $\boldsymbol{FR}_b$ in Main Rotor Axes. With unit vector $\hat{a}$ known the coning angle a_o as illustrated in Figure 83 can now be calculated:

$$a_o = \text{acos}\left(\hat{a} \cdot \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}\right) \qquad \textbf{Equation 66}$$

Note: *Calculation of the coning angle a_o as performed above assumes a flapping rigid blade with movement primarily determined by the lift force $\boldsymbol{FL}_b$ and centrifugal force $\boldsymbol{FC}_b$. Inertial, drag and coreolis forces as well as flexing of the blade were ignored. This results in a first-order model, invented on purpose to perform a gradual expansion of the* ***Main Rotor Model*** *for determining a model that is marginally good enough for use in the PRLS as described in the introduction of this Chapter.*

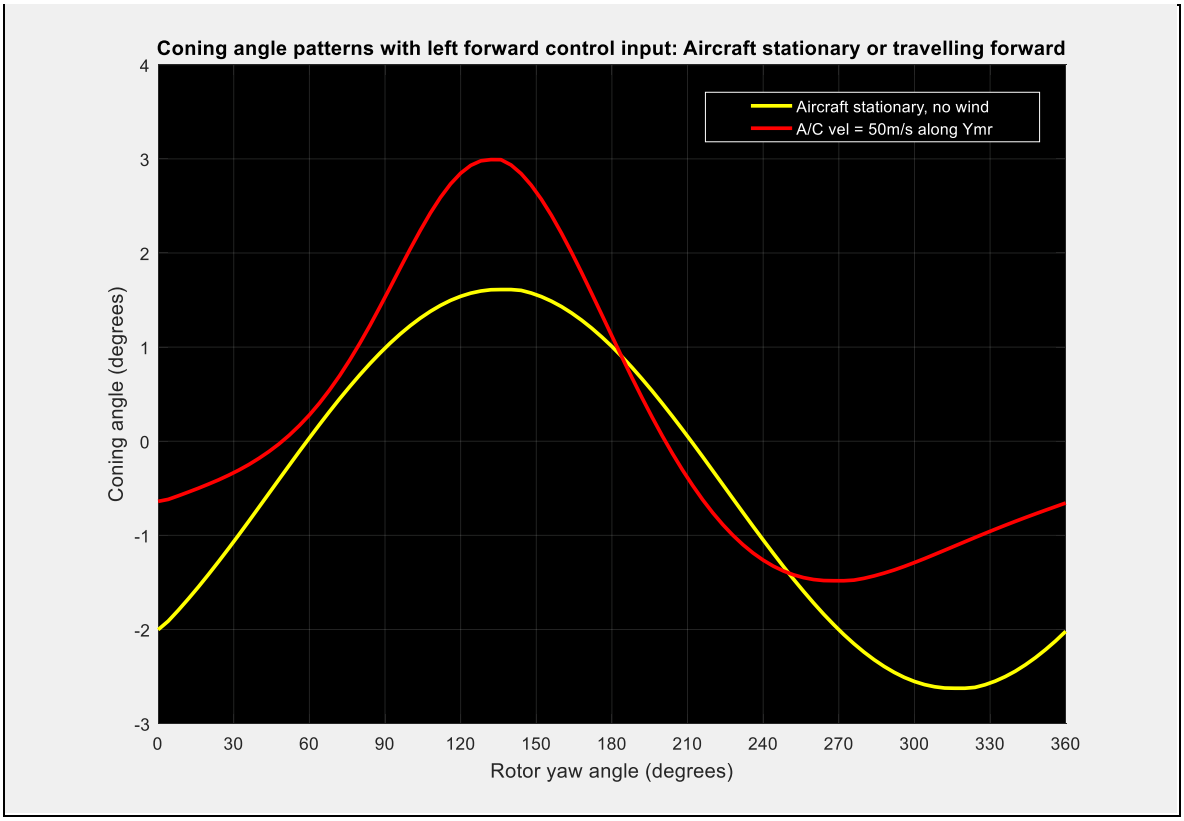


Figure 90. Coning patterns with left forward control input: Aircraft stationary or travelling 50m/sec forward [A Landman, 2016]

The coning angle of the Rotor Cone of Rooivalk AH-2A directly after application of a left-forward cyclic control input was calculated using Equation 66 for an aircraft in hover and travelling forward at 50 m/sec. The result is illustrated in Figure 90. It is evident that the the rotor cone is a dynamic entity, continually changing in shape as it responds to the positions of control levers, wind and aircraft speed.

3.5.4 Introducing Rotor Drag

Thus far, the only aerodynamic force considered for determining the coning angle of the rotor disk was the lift force $\boldsymbol{FL}_b$. This force results in a downwash that flows mainly towards the ground. However, the Main Rotor also adds a horizontal velocity component to the outflow since it drags the surrounding air with it. This factor needs to be considered by the **Main Rotor Model** because the FZ90 Rocket also exhibits high sensitivity to the horizontal component of downwash as shown in paragraph 2.5.9.2. Hence, the outflow will now be described with a vector quantity that varies in both direction and magnitude over the entire surface of the Rotor Cone.

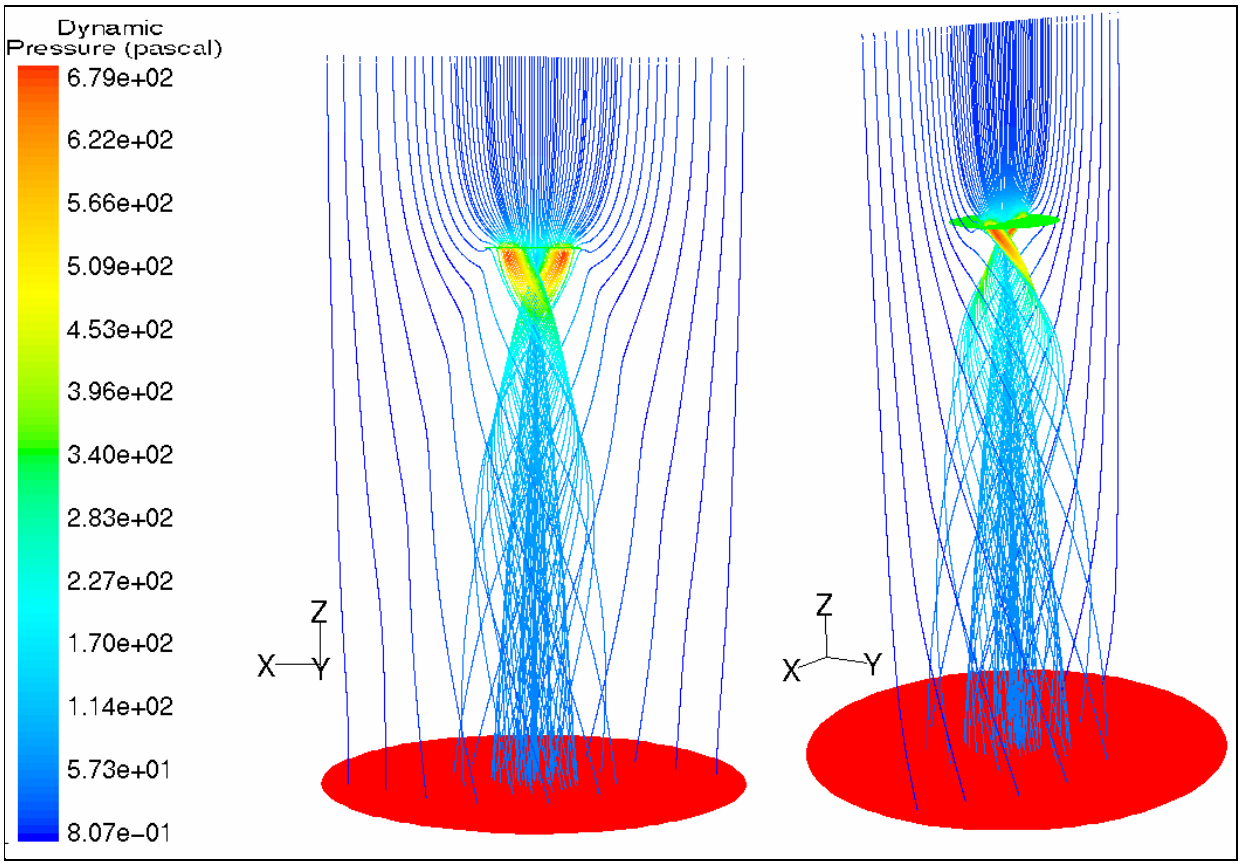


Figure 91. Helical flow in downwash as calculated with FLUENT [Chollet, 2003]

Assuming an outwash generated in accordance with the mechanism as described above, the path of an air molecule accelerated through the rotor cone should be that of a slow helix that decelerates and increases in radius as the downwash starts to expand outwards below the aircraft. An example of this kind of flow as calculated by *Chollet* [54] using FLUENT 6 is illustrated in Figure 60. Eventually the helical movement of the air molecule should stop and transform into a big vertical loop that curls back into the top of the rotor disk, accelerating as it does so. We will return to this topic in paragraph 4.5.9.

In electronic terms, the process as described above is the analogue of a 3-dimensional transmission line (i.e the air around the helicopter) that feeds back on itself through a current source (the Rotor Cone) that sources current with a specific pattern (the flux distribution of

outflow directly below the rotor cone). To predict the shape of this flow field with reasonable accuracy, the response of a large volume of ambient air surrounding the aircraft to the flux distribution over the entire rotor cone needs to be considered.

Note: *FLUENT is a Computational Fluid Dynamics program produced by ANSYS of Canonsburg, Pennsylvania in the USA.*

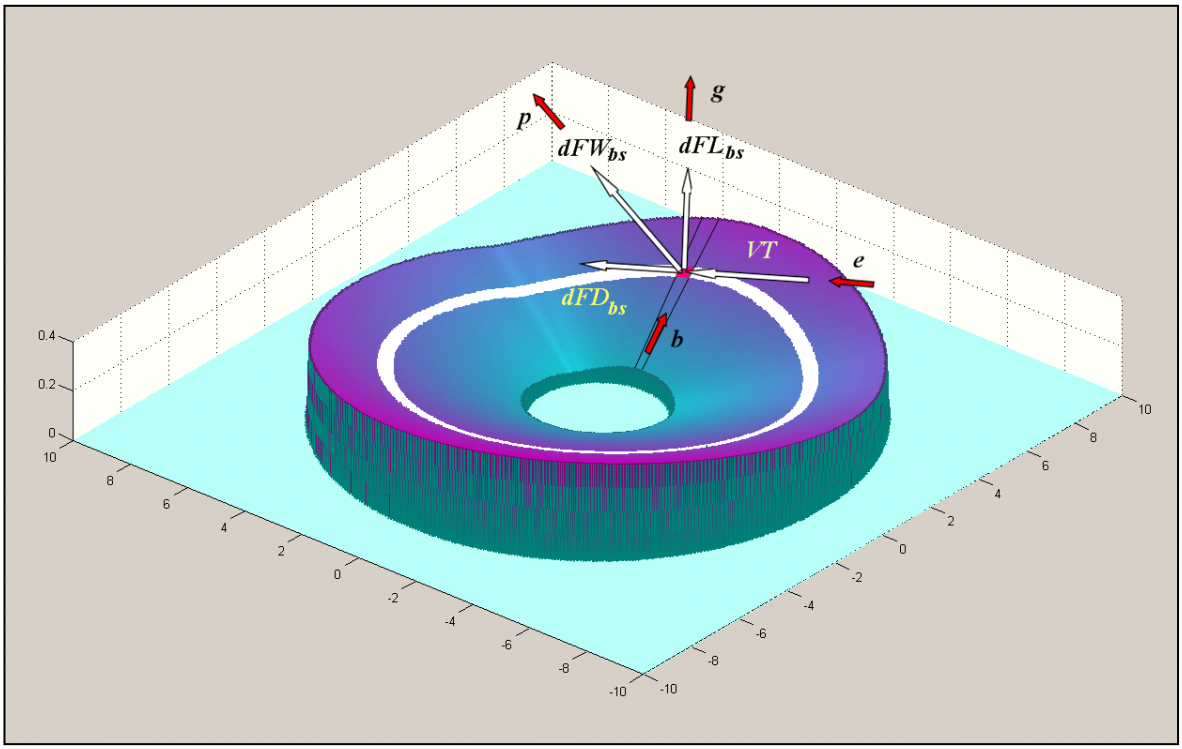


Figure 92. Drag and lift forces acting on control surface of typical rotor cone [A Landman, 2016].

Now consider a Rotor Cone as defined by the surface swept out by a flapped rigid blade following a variable path as illustrated in Figure 92. Located above the blade is a segment that corresponds with a control surface at arbitrary position as shown. The blade segment traces a variable trajectory that passes through the control surface as illustrated by the white strip in Figure 92. The blade segment generates a force $d\boldsymbol{FW}_{bs}$ composed of the vector sum of a lift force $d\boldsymbol{FL}_{bs}$ (orthogonal to vector $\boldsymbol{VT}$ and unit vector $\hat{b}$) and a drag force $d\boldsymbol{FD}_{bs}$ (parallel to vector $\boldsymbol{VT}$). While the blade sweeps through the control surface force $d\boldsymbol{FW}_{bs}$ must be generated by the acceleration of air at an oblique angle through the control surface which will thus appear to be smaller than it's normal surface area. The duty cycle of the force acting on the control surface is determined by dh in accordance with Equation 46. Note that the vectors $\boldsymbol{VT}$, $d\boldsymbol{FW}_{bs}$, $d\boldsymbol{FL}_{bs}$ and $d\boldsymbol{FD}_{bs}$ all lay in a plane which intersects the control surface and is orthogonal to unit vector $\hat{b}$.

To calculate vector $d\boldsymbol{FD}_{bs}$ note that the drag of the blade segment comprises two terms as in Equation 19. However, the equation describes the drag of a complete wing, whereas the objective here is to describe the drag of an individual blade segment. To introduce the necessary modifications note that the first term of Equation 19 is the drag of the blade segment with incident air at zero angle of attack. As described by *Clancy* [59] the drag coefficient CD_{bo} of a streamlined body such as the Rooivalk Main Rotor Blade is about 0,04 if the reference area A_o is assumed to be the frontal surface area of the blade segment. The second term of Equation 19 is the additional drag of the blade segment with incident air at non-zero angle of attack. Under these conditions, lift is being generated with concomitant energy loss that can be

described with Equation 20 if reference area A_i is assumed to be the top surface area of the blade segment. As described in paragraph 2.5.8.1 the value of the Oswald factor is typically 0,7 for a square wing. Hence, the value of e_{bs} should be 0,7 for all blade segments. Since a given blade segment does not operate on its own but as part of the overall blade, the aspect ratio of all blade segments should be as given by Equation 22 which yields a value of 11,7 for the blade used on Rooivalk AH-2A. The overall response of such a composite blade would probably be reasonably close to the real blade. Hence, combining Equation 19, Equation 20 and Equation 26 we conclude that vector $d\boldsymbol{FD}_{bs}$ can be represented with the following relationship:

$$d\mathbf{FD}_{bs} = \frac{\rho_{mr} VT^2 dr_b}{2}\left[CD_{bo}D_{bt} + \frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right]\hat{e} \qquad \textbf{Equation 67}$$

with D_{bt} the thickness of the blade, e_{bs} the Oswald factor of the whole blade and AR_b the aspect ratio of the whole blade. Combining Equation 61 and Equation 67 yields the overall force $d\boldsymbol{FW}_{bs}$ generated by the blade segment:

$$\begin{aligned} d\mathbf{FW}_{bs} &= d\mathbf{FL}_{bs} + d\mathbf{FD}_{bs} \\ &\equiv \left[\frac{\rho_{mr} VT^2 CL_{\alpha mr}\alpha_{bs} D_b dr_b}{2}\right]\hat{g} + \frac{\rho_{mr} VT^2 dr_b}{2}\left[CD_{bo}D_{bt} + \frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right]\hat{e} \end{aligned} \qquad \textbf{Equation 68}$$

Equation 68 represents force $d\boldsymbol{FW}_{bs}$ calculated in terms of aerodynamic considerations. However, it can also be calculated in terms of momentum considerations as reflected by the rocket equation as in Equation 1. In such a case the control surface can be seen as an orifice (jet) through which the air is imparted a velocity change (acceleration) antiparallel to force $d\boldsymbol{FW}_{bs}$. To use the rocket equation for calculating the force generated by the current control surface (orifice), the total massflow into the current control surface is required. This is given by the product between inflow air density, velocity of inflow normal to the current control surface and the surface area of the current control surface. To calculate this product define unit vector $\hat{p}$ that points in the same direction as vector $d\boldsymbol{FW}_{bs}$ so that:

$$\begin{aligned} \hat{p} &= \frac{d\bar{F}\bar{W}_{bs}}{dFW_{bs}} \\ &\equiv \left\{\frac{\left(CL_{\alpha mr}\alpha_{bs}D_b\right)\hat{g} + \left[CD_{bo}D_{bt} + \frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right]\hat{e}}{\sqrt{\left\{CL_{\alpha mr}\alpha_{bs}D_b\right\}^2 + \left\{CD_{bo}D_{bt} + \frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right\}^2}}\right\} \end{aligned} \qquad \textbf{Equation 69}$$

so that:

$$\begin{aligned} \dot{m}_{cs} &= \rho_{mr} A_{cs}\left(\hat{g}\cdot\mathbf{V}in\right) \\ &\equiv \rho_{mr}\left[R_f + r_b\cos\left(a_o\right)\right] dh\, dr_b\left(\hat{g}\cdot\mathbf{V}in\right) \end{aligned} \qquad \textbf{Equation 70}$$

with $\dot{m}_{cs}$ the total massflow into the current control surface. Application of the equivalence of force principle now yields the following relationship:

$$\frac{\rho_{mr} VT^2 dr_b}{2}\left\{\left[\frac{CL_{\alpha mr}\alpha_{bs} D_b}{2}\right]\hat{g}+\left[CD_{bo}D_{bt}+\frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right]\hat{e}\right\}\left(\frac{2dh}{\pi}\right)=$$

$$\left\{\frac{\left(CL_{\alpha mr}\alpha_{bs}D_b\right)\hat{g}+\left[CD_{bo}D_{bt}+\frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right]\hat{e}}{\sqrt{\left\{CL_{\alpha mr}\alpha_{bs}D_b\right\}^2+\left\{CD_{bo}D_{bt}+\frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right\}^2}}\right\}\rho_{mr}\left[R_f+r_b\cos\left(a_o\right)\right]dh\, dr_b\left(\hat{g}\cdot\mathbf{V}in\right)dV \qquad \textbf{Equation 71}$$

with dV the amplitude of velocity change through the current orifice. Re-organization of Equation 71 and noting that the velocity change must be antiparallel to unit vector $\hat{p}$ yields:

$$d\mathbf{V}=-\hat{p}\left[\frac{VT^2\sqrt{\left\{CL_{\alpha mr}\alpha_{bs}D_b\right\}^2+\left\{CD_{bo}D_{bt}+\frac{\left(CL_{\alpha mr}\alpha_{bs}\right)^2 D_b}{\pi\, e_{bs} AR_b}\right\}^2}}{\pi\left[R_f+r_b\cos\left(a_o\right)\right]\left(\hat{g}\cdot\mathbf{V}in\right)}\right] \qquad \textbf{Equation 72}$$

with $d\boldsymbol{V}$ the velocity change through the Rotor Cone at the control surface. Note that $d\boldsymbol{V}$ comprises three components – the first due to lift generation (generally pointing towards ground), the second due to Main Rotor drag (generally pointing in a horizontal direction) and the third a modified remnant of the inflow velocity. Finally the outflow velocity $\boldsymbol{V}_{out}$ is given by the following:

$$\mathbf{V}_{out}=\mathbf{V}_{in}+d\mathbf{V} \qquad \textbf{Equation 73}$$

Note that $\boldsymbol{V}_{in}$ and $\boldsymbol{V}_{out}$ differ both in direction and amplitude.

3.5.5 Modelling the Main Rotor flow

The air flow of the Main Rotor of Rooivalk AH-2A as observed in Main Rotor Axes was calculated with Equation 49 to Equation 72. The results of the simulations for typical scenarios during which rockets can be delivered from Rooivalk AH-2A are illustrated in Figure 93 to Figure 96. Note that these figures represent momentary outflow distributions directly below the Main Rotor with inflows held homogenous and constant to make possible the simulations as shown. When combined with the **Space** block of Chapter 4 the inflow becomes a dynamic entity with variable flow distribution that is strongly affected by flight conditions.

.

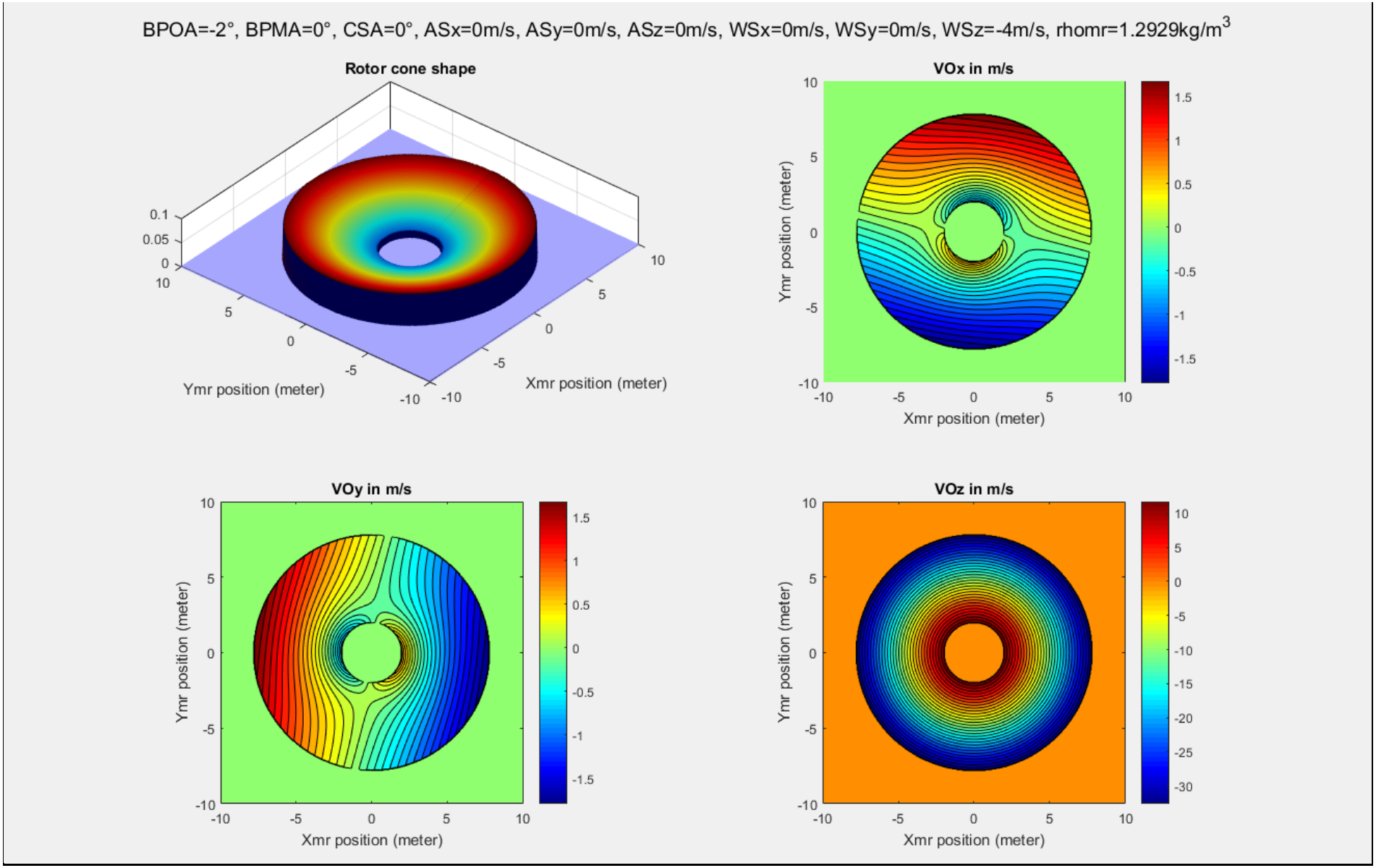


Figure 93. Outwash of Rooivalk AH-2A during hover [A Landman, 2016]

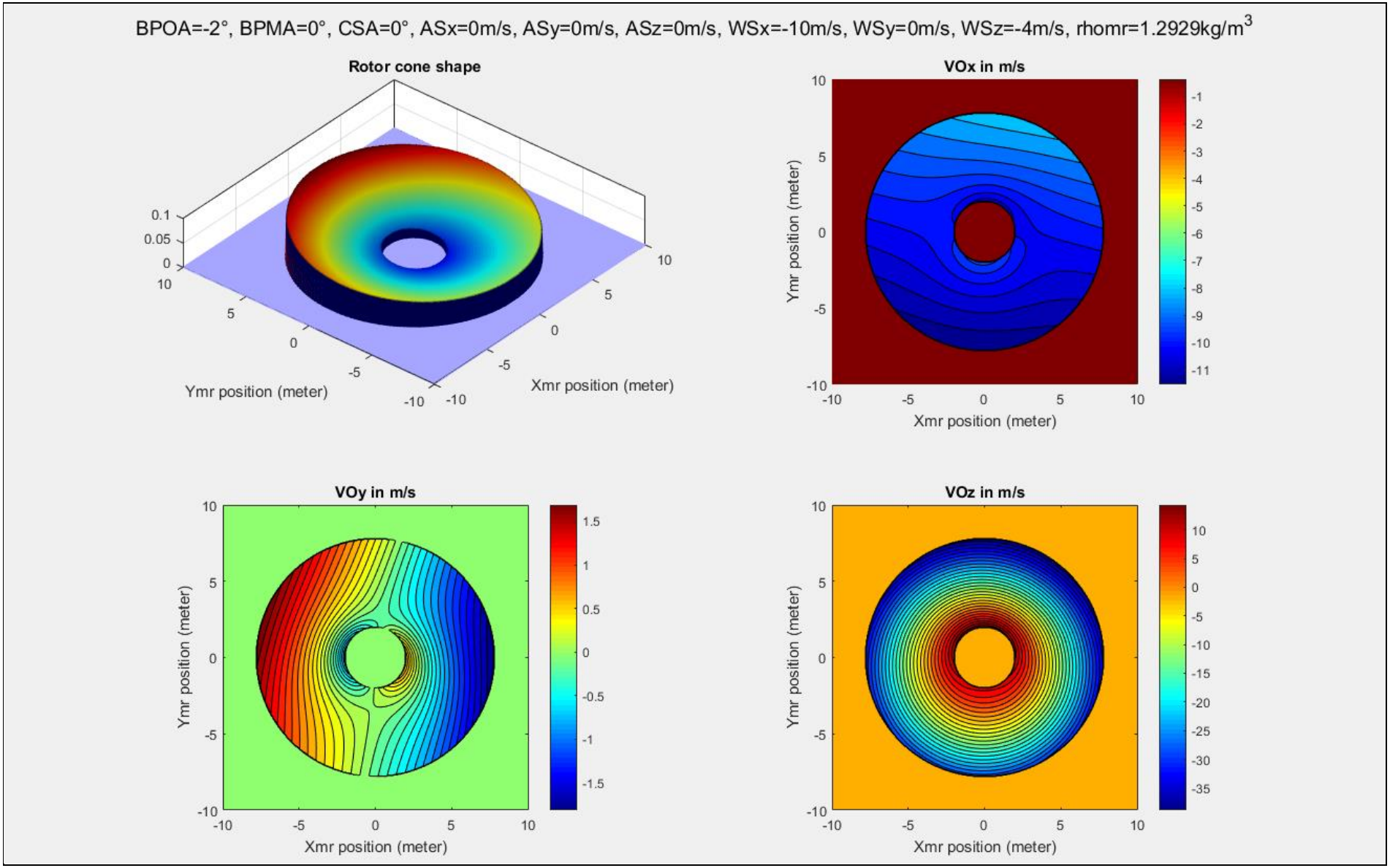


Figure 94. Outwash of Rooivalk AH-2A during hover with wind from the right [A Landman, 2016]

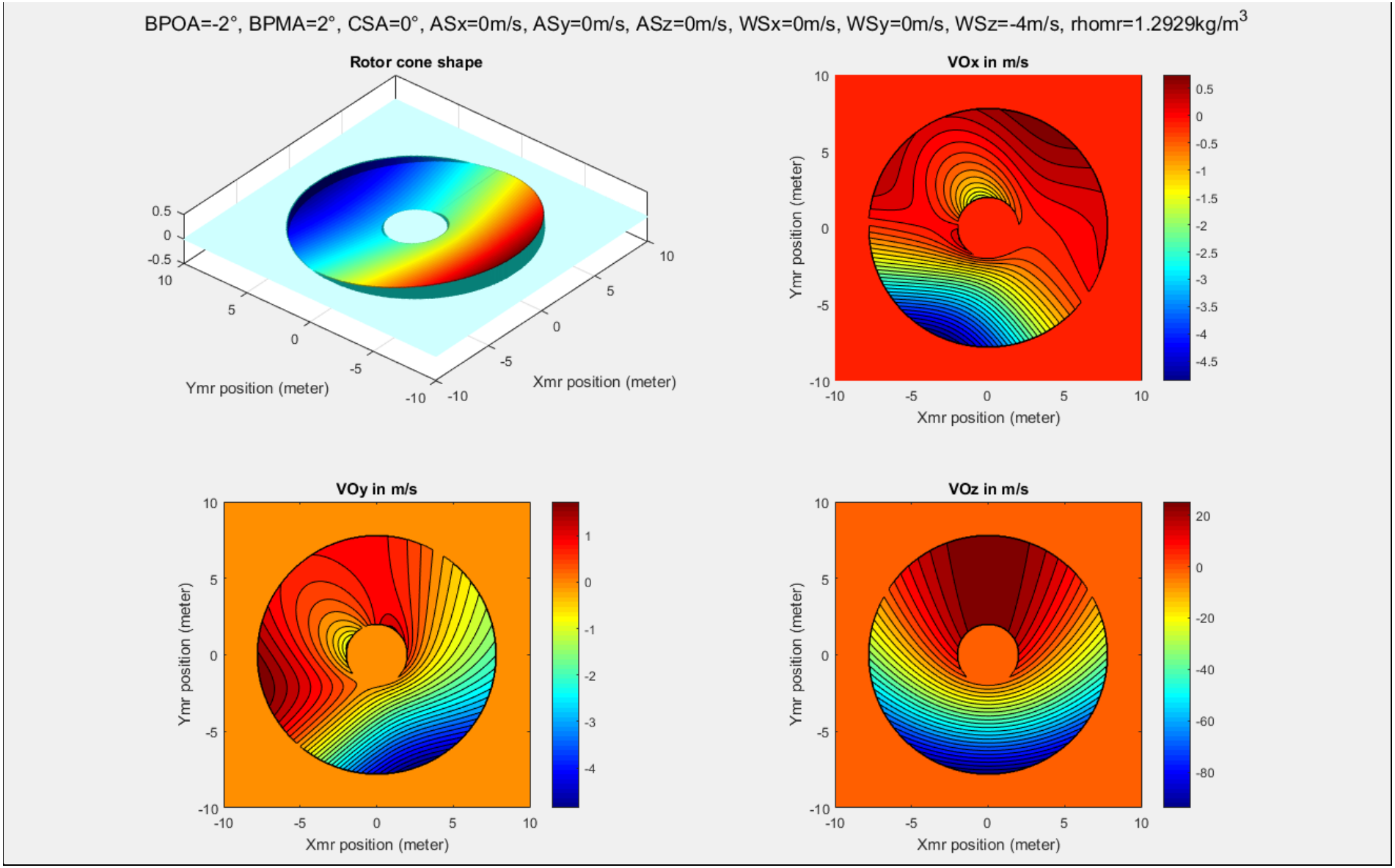


Figure 95. Outwash of Rooivalk AH-2A during transition from hover to forward flight [A Landman, 2016]

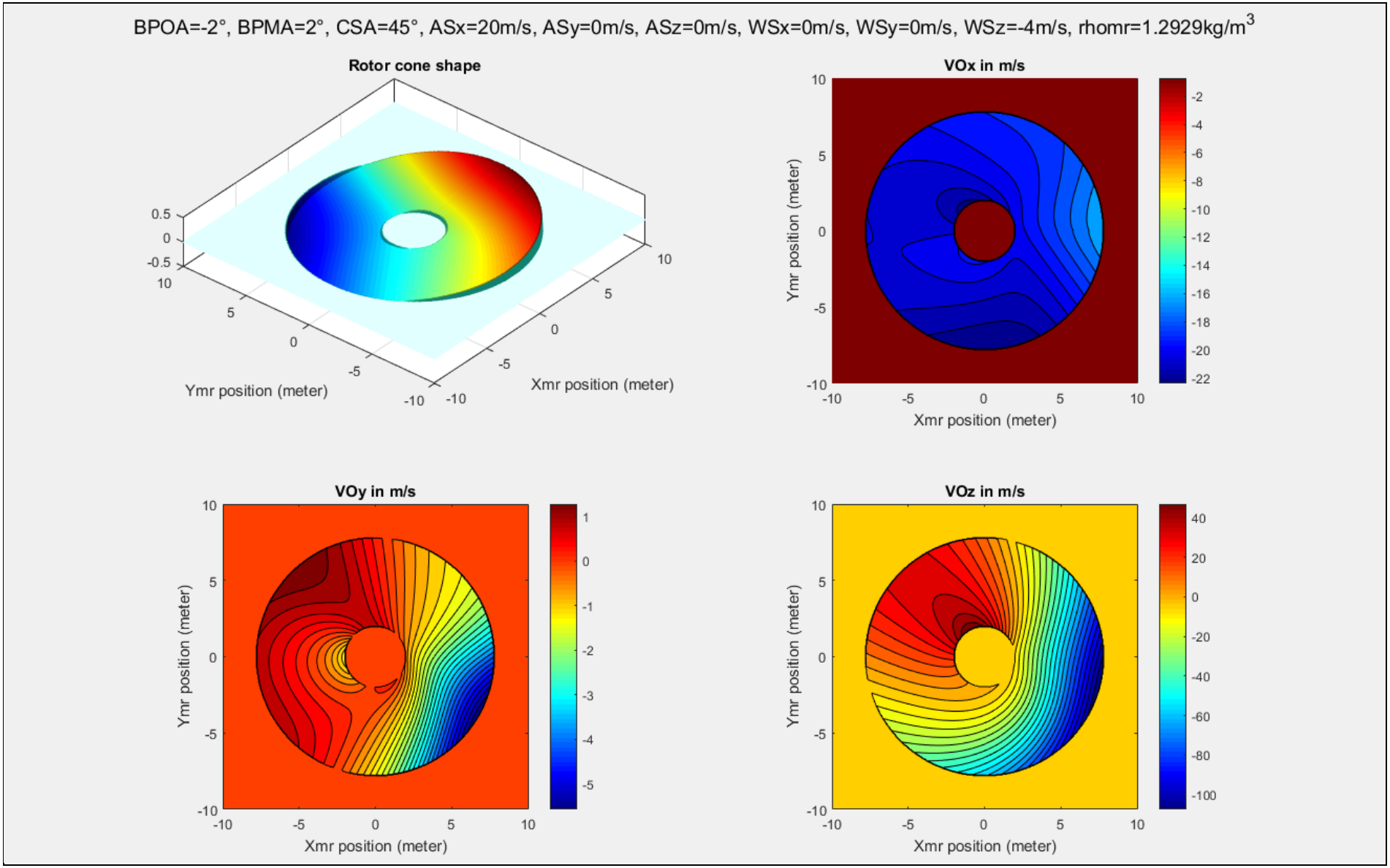


Figure 96. Outwash of Rooivalk AH-2A during application of left-forward cyclic during forward flight [A Landman, 2016]

Note: *Assuming the change in air density above and below the Main Rotor Disk to be negligibly small, Figure 93 to Figure 96 essentially portray the Main Rotor as a so-called Current Dependant Current Source in electrical circuit analysis terms. For the purposes of the calculation here each case was simulated by holding the inflow constant (which allows for omission of the* ***Space*** *block as illustrated in Figure 20). In reality the inflow is determined by some external influences and outflow itself as modificied by the* ***Space*** *block. Note that this implies a feedback with variable time delay which can introduce a 180° phase shift which can result in instability (flutter). Investigation of the* ***Space*** *block is the topic of Chapter 4.*

3.6 Summary (Develop general theories)

This chapter was used to initiate the third Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. As shown in Chapter 2 the downwash of the helicopter must be calculated with good enough (not perfect) accuracy and in real time for the PRLS concept to work. The objective of this chapter was to develop a **Main Rotor Model** that would be (just) good enough to achieve the PRLS accuracy as postulated in Chapter 1.

The primary observational evidence presented for developing the **Main Rotor Model** included the mechanical layout of the Main Rotor as used on Rooivalk AH-2A, the Actuator Disk Theory of *Froude* [53] and the fact that viewed sideways the main rotor of a helicopter seems to form a virtual cone which changes shape depending on flight conditions and what the pilot does with his controls. Also presented were concepts from electrical circuit analysis which are analogues to the fluid flow mechanisms as presented in this chapter. These include transmission lines (in which all disturbances travel with finite speed, superimpose on each other and influence the entire network) as well as current sources (which source constant currents through their terminals irrespective of the voltages present at these terminals).

Questions were asked as to what type of model would satisfy the requirement to model the main rotor of a helicopter in real time and with good enough accuracy. From the research performed in Chapter 2 it was evident that the FZ90 Rocket is very sensitive to downwash along any axis. For this reason the **Main Rotor Model** had to be such that it would excite large and small flows of the downwash flow pattern that can have an influence on the trajectory of the FZ90 Rocket. As described by various researchers such as *Kokalari et al* [28], *Woodson & Melcher* [29] and *Karnopp, Margolis & Rosenberg* [31], the size of these local flows were estimated to be an order of magnitude smaller in size than the length of the FZ90 Rocket, based on experience. This meant that as a minimum, the flow pattern over the Main Rotor Disk had to be modelled as a 3-dimensional flow field versus the 2-dimensional flow field as in Blade Element Theory originally formulated by *Froude* [53]. Note that this conclusion also means that objects such as fuselage, ground and buildings need to be considered in calculating downwash (this is one of the objectives of Chapters 4 and 5).

Exploiting the observational evidence as described above the Floppy Disk Hypothesis was formulated. This hypothesis states that the sophistication required to model with good enough fidelity the flow through the main rotor of a helicopter can be achieved by representing the main rotor as a very thin 3-dimensional surface of current controlled current sources which morphs into various shapes and velocity distributions as a function of flight conditions and flight control inputs.

The prediction made with the Floppy Disk Hypothesis was that the sophistication of the model could be expanded in steps until a point would be reached where the accuracy of the model is good enough for the purposes of the PRLS based on subjective judgement.

Persuing the Floppy Disk Hypothesis the **Main Rotor Model** was developed in this chapter. This model makes some major assumptions. First, it assumes the blades of the Main Rotor to

be rigid, flapped and capable of being adjusted in pitch as function of Swash Plate position. Second, it assumes the orientation of the blades to be dominated by an equilibrium between centrifugal force and lift, thus sweeping out a variable (floppy) surface referred to as the Main Rotor Cone. Third, it sub-divides the Main Rotor Cone into a fine mesh with each sub-division (control surface) behaving like a thin jet (orifice) that obeys the Rocket Equation. Fourth, it assumes the blade to generate the ordinary aerodynamic forces of lift and drag as it sweeps the Main Rotor Cone. Fifth, it assumes that with the blade underneath a given orifice the lift and drag of the blade in that location is generated by the inflow into the orifice being accelerated by the blade such as to produce the combined force as observed.

Expanding the **Main Rotor Model** in steps to consider mechanical force equilibrium, lateral dyssymetry and drag has proved the basic assertion of the Floppy Disk Hypothesis that the model can be expanded peacemeal. The outflow of the Rooivalk AH-2A Main Rotor was calculated with the **Main Rotor Model** for a number of load cases. To do this the inflows were clamped at homogenous conditions that approximate each situation. Even so, the experiments yielded higly variable outflows which in the final system will feed back via Space to generate a complex non-homogenous inflow. The downwash of a helicopter is indeed a very complicated thing.

The main contributions of this chapter was development of the **Main Rotor Model** based on a combination of a 3-dimensional version of Blade Element Theory, the rocket equation and standard theory of aerodynamic forces of lift and drag. Indications are that this model will probably require one further increase in sophistication in order to model the wakes generated by each main rotor blade (which are sure to affect the trajectory of the FZ90 Rocket). This will fit in well with the Commutation Process as conceived in Chapter 5. In conceiving the **Main Rotor Model** we have shown that such a model cannot be used in isolation, but that it has to be integrated with the **Space** block of Figure 20 as a minimum.

4 The Gas Flow Model (Cycle 4)

"*Big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity*". Lewis Fry Richardson.

This chapter introduces the fourth Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. In order to explain the observed coning behaviour of rockets launched from a helicopter Chapter 2 formulated the Coning Hypothesis. After proving the Coning Hypothesis, Chapter 2 concluded that the downwash of the helicopter would have to be calculated with high accuracy and in real time for the PRLS concept to work.

The purpose of this chapter is to conceive the **Space** block of the **Rocket Launching System Model** as illustrated in Figure 20, such that it is (just) good enough to achieve the PRLS accuracy as postulated in Chapter 2. This will be done by attempting to conceive an algorithm both suitable for modelling the flow pattern of a helicopter (as excited by the **Main Rotor** block) and suitable for implementation with massively parallel hardware.

Observational evidence of the shape of the downwash of a helicopter is presented in the form of a number of photographs depicting the flow field using water droplets or dust particles released into the downwash during so-called brownout conditions as illustrated in Figure 97. Additional observational evidence in the form of Light Detection and Ranging (LIDAR) measurements of the shape and size of a helicopter downwash is also presented.

Figure 97. Brownout generated by EH-101 during IGE hover flight [LZDZ, 2017]

Taking note of the fact that the photographic evidence shows the downwash to be in the form of a donut that elongates to the rear depending on speed, questions are asked as to what the mechanisms involved in creating the downwash of a helicopter might be. This results in the Donut Hypothesis which states that the downwash is analogous to the response of an electrical circuit in the form of a 3-dimensional transmission line (space filled with air bent on concerving the laws for the conservation of energy and mass) being excited by a current source (main rotor acting as a pump).

Under the Donut Hypothesis it follows that Space and the Main Rotor can be modelled as two separate entities exchanging information through a suitable interface mechanism (referred to as *commutation* for the purposes of this study). It also means that the downwash follows the path of least resistance, thus forming the donut shape as observed. A set of equations that describe this mechanism was first formulated by *Navier* in 1822 and improved later by *St Venant*, *Poisson* and *Stokes* [63]. These equations became known as the Navier Stokes equations. Here they will be casted in a special form referred to as the *Taylored Navier Stokes Equation* for the purposes of this study. The form of the *Taylored Navier Stokes Equation* lends itself for implementation on a dedicated supercomputer (referred to as the *Array Processor* for the purposes of this study and developed in later Chapters of this study).

To test the Donut Hypothesis a finite difference model referred to as the **Downwash Model** is developed to simulate Space. Whereas Space was modelled as a relatively simple entity with few Degrees of Freedom (DOF) in Chapter 2, it is modelled with millions of DOF in the **Downwash Model** to achieve the accuracy required for the PRLS. A solver that is both simple (to maximize speed of execution) and stable in both the time and spatial domains is developed. By coding this model in Matlab, the *Taylored Navier Stokes Equation* is shown to generate reasonable results for a number of test cases.

Finally, the **Main Rotor Model** as developed in Chapter 3 is used to stimulate a collection of **Downwash Model** blocks configured for different sizes of Computational Volumes or so-called *Flight Boxes*. The minimum size of the Computational Volume as required by the PRLS is determined by evaluating the resulting downwash flow patterns. This minimum Flight Box size forms an important driver in subsequent chapters of this study.

The main contributions to the knowledge database made in this chapter is determination of the Flight Box size and formulation of the concept that the downwash of a helicopter can be modelled with three discrete blocks referred to as the **Taylored Navier Stokes** block, **Commutation** block and **Probing** block. This concept lays the foundation required in later chapters to package a supercomputer in limited volume and overcome the limitations of Amdahl's Law .

4.1 Donuts and Tails **(Make observations)**

Some indications of what the downwash of a helicopter might look like are illustrated in Figure 97, Figure 98, Figure 99, Figure 100 and Figure 101. The photographs of Figure 97, Figure 98 and Figure 99 seem to suggest that for a helicopter hovering under In Ground Effect (IGE) conditions the downwash has the shape of a donut while for a helicopter in IGE forward flight it has the shape of an elongated donut. In fact, for a helicopter in OGE forward flight the donut morphs into a downward tail behind the helicopter, similar to that of a fixed wing aircraft.

Figure 100 and illustrates the test setup as used by *Sjöholm et Al* [61] to measure the downwash of a SH-3 Sea King helicopter using two Light Detection And Ranging (LIDAR) sensors. Figure 101 illustrates the wind speed in a vertical plane directly below the helicopter with the helicopter hovering 21 meter above ground as measured by *Sjöholm et Al* [61].

While this is the general downwash flow pattern of a helicopter it is by no means the only one. There are in fact various flow patterns, some totally counter-intuitive depending on flight conditions. *Coyle* [60] describes a "Low Altitude Vortex" that forms in front of the helicopter and moves towards the rear of the aircraft during transition from hover to forward flight. Close to hover this vortex is located in front of the helicopter where it is sure to influence a passing FZ90 Rocket. *Coyle* [60] also describes other strange conditions such as air flowing upwards through an area of reversed flow that appears in the main rotor disk during forward flight.

Inspection of Figure 98 shows that the downwash of a helicopter is not homogeneous as assumed in paragraph 2.5.8.4. Instead, it is characterized by rapid variations in density and speed seemingly caused by individual main rotor blades. Inspection of Figure 98 and Figure 99

shows that for a helicopter in forward flight the donut morphes into two inward vortices that ultimately combine into a single downstreaming rear end.

Figure 98. Water spray generated by Apache during IGE forward flight [Special Ops Magazine, 2014]

Figure 99. Downwash of R-50 morphing into two downward tail at rear [Yamaha, 1987]

It is clear that depending on the flight conditions a rocket being launched from a helicopter may encounter a variable and high-intensity downwash directly below the main rotor during launch

initiation followed by a lower intensity upwash as it moves away from the aircraft. Within these main areas smaller high-intensity flows and swirls will cause a cacophony of different weather-cocking forces acting on the rocket. If the helicopter is travelling at high enough speed the rocket may not encounter any of these effects at all.

Figure 100. Using two LIDAR's to measure air velocity below SH-3 Sea King helicopter [Sjöholm et al, 2014]

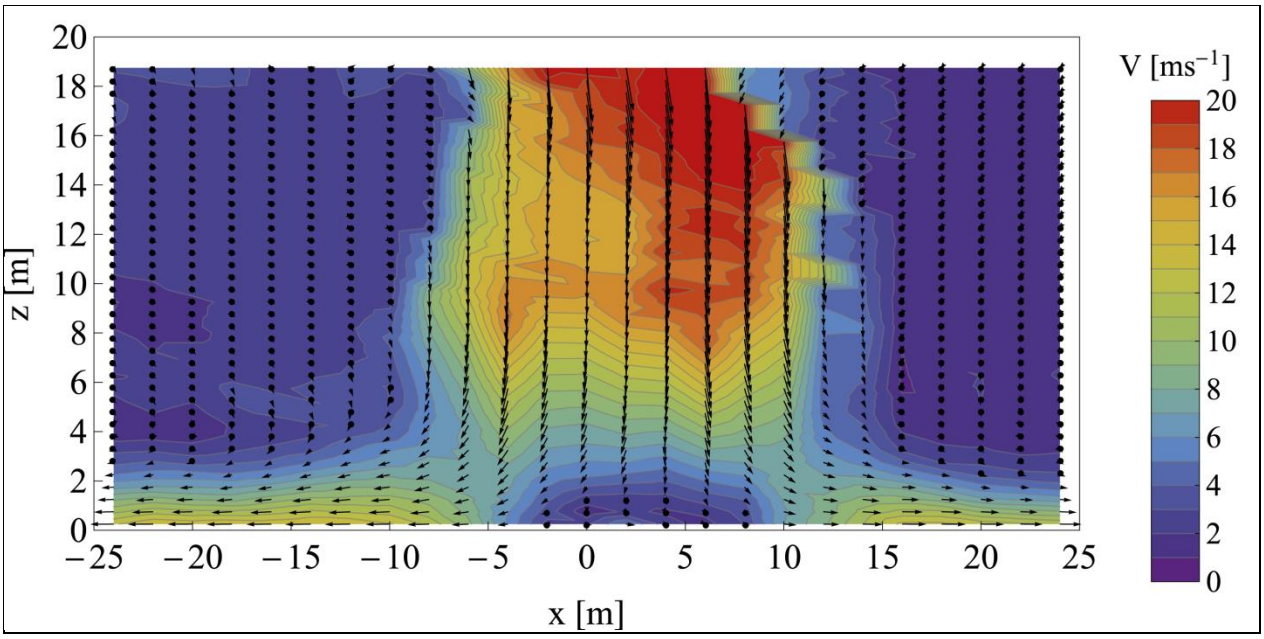


Figure 101. Measured velocity distribution in vertical plane below SH-3 Sea King helicopter [Sjöholm et Al , 2014]

4.2 Why a donut? (Think of interesting questions)

Why would the downwash of a hovering helicopter be in the form of a donut? This question can be answered by noting that air is a fluid that complies with the laws of conservation of energy, momentum and mass. In addition, air is a gas that reacts in accordance with an equation of state such as the universal gas law.

Viewed in electrical terms the space around the helicopter can be seen as a 3-dimensional transmission line that is activated by a current source (the main rotor) at centre of the 3-dimensional transmission line. In such a system the current will follow the shortest route, inducing voltages in the transmission line that wil cause the current to spread into the form of a donut as observed.

The flow of air in space is analogue to the flow of electricity in a transmission line because both mediums share similar characteristics. Current flowing in an inductor resembles energy stored as a magnetic field, similar to energy stored as momentum in moving air. Charge stored in a capacitor resembles energy stored as an electric field, similar to energy stored in compressed air. In the electrical case current must be conserved, in the flow field matter must be conserved, and so on. As described by *Moore* [62] the governing equations of a transmission line are known as the Telegraphers Equations. As described by *Anderson* [45] and *Rogers* [63] the governing equations of a fluid flow field are known as the Navier Stokes Equations.

4.3 The Donut Hypothesis (Formulate Hypothesis)

Having considered the observational evidence, we conclude that the downwash flow field of a helicopter is governed by the Navier Stokes equations with boundary conditions dictated by various bodies such as the main rotor blades, fuselage and ground. This downwash flow field can be predicted to any degree of accuracy by solving the Navier Stokes Equations with suitable boundary conditions imposed and taking suitably small steps in space and time. In addition, the Navier Stokes Equations can be written in a form conducive for implementation on a dedicated computer, such that the equation can be solved in real time, defined as an iteration rate in excess of 10 kHz for the purposes of this study.

4.4 Donut Predictions (Develop Testable Predictions)

Some predictions based on the Donut Hypothesis can now be made:

- A version of the Navier Stokes equation can be derived that would be suitable for dividing the downwash problem into many parts (test volumes) that can all be executed in parallel.
- The test volumes as mentioned above can be implemented in sequential fashion on a von Neumann machine (yielding a slow process) and such a simulation would confirm the donut shape of the downwash of a hovering helicopter.
- The simulation as described above would indicate a diameter of about 100 meter as observed for the downwash of a helicopter in the 8 ton class. This dimension can be estimated from Figure 97 which illustrates the shape of the EH-101 downwash during In Ground Effect (IGE) conditions. Inspection of the photograph shows that the EH-101 downwash is about five aircraft lengths in diameter. Since the longitudinal length of EH-

101 is 22,77 meter (front of main rotor edge to rear of tail rotor edge), the diameter of it downwash flow field under IGE conditions must be about 114 meter. Note the EH-101 has an empty weight of 10,5 ton and a maximum takeoff weight of 14,6 ton. Hence, the downwash of a helicopter in the 8 ton class should be a little less – probably about $\left(\frac{8}{10,5}\right) * 114 \approx 90$ meter.

- The maximum wind speed in the downwash of Rooivalk AH-2A should be about 20 m/s. This prediction is based on the downwash measurements as made by Sjöholm et Al [61] using a Sea King helicopter and illustrated in Figure 101.

4.5 Building the Downwash Model (Gather Data To Test Predictions)

Predicting the downwash of a helicopter is the topic of a great number of papers and studies from the open literature, most of which use the Navier Stokes equations to predict the complex interaction between air, rotors and other objects such as the fuselage and ground. For most of these studies accuracy is of prime importance - the models run a-priori and are not intended to be real time applications. Where wall clock time is considered it is only relevant so far as ensuring the simulation has reasonable computational turnaround time that is usually measured in hours or minutes but certainly not in milli-seconds.

Thibault and *Senocak* [64] describe the use of the Compute Unified Device Architecture (CUDA) language in combination with multiple Graphic Processor Units (GPU) to solve the so-called lid-driven cavity problem using the Navier Stokes equation. They report a speedup (i.e. a relative comparison) of 100 times using the NIVIDIA S870 Tesla server fitted with four GPU's compared to the AMD Opteron processor running at 2.4 GHz. Although their objective was clearly an improvement in speed of computation, their simulation probably required a wall clock time measured in hours to complete.

Chandar, *Sitaramany* and *Mavriplis* [65] describe a similar application in which use was made of a combination of Central Processor Units (CPU) and GPU's to solve the flow field of a so-called Caradonna and Tung rotor. They report a wall clock time as low as 0,4 seconds to complete 10 iterations.

Bludau, *Rauleder*, *Friedmann* and *Hajek* [57] describe the use of a NVIDIA GPU graphics board in a helicopter training simulator where the interaction between the downwash flow field (what they call INFLOW) and main rotor as controlled by the Pilot must be modelled in real time. They do not state any specific performance figures other than to say that the system is "Real-Time Ready".

No paper from the open literature could be found on a gas flow solver running even close to 10 kHz. The closest were some papers by *Stam* [67] [68] [69] on gas flow solvers intended for use in computer games. *Stam* does not quote any quantitative performance figures because his algorithm is intended as a building block for computer games (which call the block as required) but it is clear that his model would have to run at a minimum rate of about 16 iterations/second in order for the image to evolve without jitter.

Many fluid flow models make assumptions which cause simplification of the Navier Stokes Equations. These include assuming the gas to be incompressible [66] [57] [64] [67] [68] [69], re-arranging the Navier Stokes equations to yield the so-called Thin Layer Navier Stokes equations [70] or estimating the viscous fluxes in terms of average flows to yield the so-called Reynolds Averaged Navier Stokes (RANS) equations [70]. Assuming the fluid to be inviscid (i.e. so thin as to have negligible viscosity) leads to the so-called Euler Equations [70].

In the case of computer games the requirement for modelling gas flow is that it must "look right". This allowed *Stam* [67] to introduce radical adaptations such as using the Fast Fourier

Transform and Inverse Fast Fourier Transform to model the viscous effects of a fluid through multiplication with a suitable filter in the frequency domain. Note that this implies convolution in the spatial domain. This is equivalent to the scenario applicable to this study where accuracy can be traded for speed of operation, since it makes no sense to calculate the downwash to such detail where other processes such as oscillation of the Stub Wing or variability of ambient wind start to dominate the ballistic performance of the system.

Although accuracy was a tradeable requirement in the case of this study, what and how to perform such a trade-off was not known at this stage. The approach was therefore to make no assumptions to simplify the Navier Stokes Equations other than assuming conductive heat flows due to temperature differences to be negligible (this is a reasonable assumption since the temperature change induced in the air as it moves through the rotor of a helicopter is relatively small). To introduce the (radical) characteristics necessary to achieve real time operation the power of massive parallelism offered by contemporary electronics was pursued instead. To do that the Navier Stokes equations were derived from basic principles [45] [63] and written in a form suitable for implementation on Field Programmable Gate Array (FPGA) and conducive for exploiting the features of the equation itself (e.g. simplifications based on zero terms in sparse matrix).

Another characteristic to be considered is the fact that the Navier Stokes equation is inherently unstable. This fact was discovered by *Lewis Fry Richardson* when he tried to perform a weather forecast for 20 May 1910 based on numerical methods [71] [72]. The forecast was a failure caused by noise in weather measurements that were used as initial conditions for *Richardson's* model. When the high frequencies in the data were later removed by researchers using a Chebyshev filter, Richardson's model proved remarkably correct. The technique of rendering the weather equations stable by removing unwanted high frequencies through filtering would later be developed by *Jule Charney* [72].

The version of the Navier Stokes equation as developed in this study is of the form $\frac{\partial \mathrm{X}}{\partial \mathrm{t}} = [\mathrm{A}]^{-1}[\mathrm{B}]$ with **X** a 1x5 matrix representing the state variables (pressure, temperature and velocity components) of a given test volume, **A** a 5x5 matrix dependant only on the current value of **X** and **B** a 1x5 matrix dependant on the spatial second partial derivatives of **X**. Our initial attempts to integrate this differential equation with the relationship $\mathrm{X_{n+1}} = \mathrm{X_n} + \frac{\partial \mathrm{X}}{\partial \mathrm{t}} * \Delta \mathrm{t}$ revealed instability that could not be solved with any amount of filtering. This same type of instability was also found by *Foster* and *Metaxas* [73] and solved by *Stam* using his "Stable Fluids" method [69]. Note that Stam's method is a so-called semi-Lagrangian technique already developed by Courant et al [74] during the 1950's.

The Stable Fluid method of *Stam* [69] involves tracing the current state variables back to their values a time $\Delta \mathrm{t}$ ago and using these values to derive a new estimate based on extrapolation and summation between adjacent test volumes. From a control perspective this method seems to represent a lead compensator operating from the past, introducing a phase shift such as to render the algorithm stable in current time. In the case of this study the same problem of instability was solved by using a Taylor series to estimate the value of $\mathrm{X_{n+1}}$ with the Taylor series based on $\frac{\partial \mathrm{X}}{\partial \mathrm{t}}$ and the value of $\mathrm{X_{n-1}}$. In other words, a moderation of the value of $\frac{\partial \mathrm{X}}{\partial \mathrm{t}}$ based on history similar to the method of *Stam*.

A further problem of instability as detected in this study occurred when the maximum frequency of the gas flow solution exceeded the Nyquist frequency [19] of the algorithm (i.e. $\mathrm{f_{nt}} = \frac{0,5}{\Delta \mathrm{t}}\ Hz$) in the temporal domain. Similar problems were encountered when the maximum frequency of the gas flow solution exceeded the Nyquist frequency of the algorithm $f_{ns} = \frac{0,5}{\Delta x}\ cycles/meter$ in the spatial domain. Under these conditions aliases appear that render the simulation useless. The solution adopted in this study was to eliminate all frequencies above Nyquist by introducing a recursive filter in the time domain and a moving average filter in the spatial domain.

For calculating the perturbation of the FZ90 Rocket due to downwash the iteration rate of the solver is therefore determined by the time taken for the tip of a main rotor blade to transit a single test volume. For Rooivalk AH-2A and a grid size of 121 mm (as derived below) this time is about 0,36 msec which means the time step Δt must be smaller than 0,18 msec and the iteration rate larger than 5,5 kHz. Choosing an over-sampling factor of 2 sets the required iteration rate at 10 Khz.

In similar fashion the minimum wavelength of the gas flow solution in the spatial domain should be about an order of magnitude smaller than the length of the FZ90 Rocket (to detect the torques acting on the Rocket due to differences in wind velocity along the length of the FZ90 Rocket body). Since the FZ90 Rocket is about 1,2 meter in length the grid size should be 121 mm or shorter.

4.5.1 Basic approach and conventions

To derive a relationship for describing the downwash of a helicopter, imagine a so-called Flight Box as illustrated in Figure 102. The Flight Box is the computational domain of the problem - a moving region in space with finite dimensions centred around the helicopter. It sweeps through space in unison with the helicopter and contains a large number of test volumes that are stationary in Inertial Axes. These test volumes are numerous enough to contain the entire downwash pattern such that the speed of air at the edges of the Flight Box approaches the ambient wind conditions.

Next imagine the test volumes to have dimensions Δx, Δy and Δz which cause each test volume face to have surface area A as illustrated in Figure 103. Encased within the Flight Box are objects (moving and stationary) such as the Aircraft, Main Rotor, Ground and Rockets so that some test volumes are filled with air and others with other materials such as soil, metal, water, etc. In this model, each object moves by vacating a set of test volumes and occupying an adjacent set of test volumes corresponding with the geometry and movement of the particular body. Similarly, a large number of test volumes are occupied by the Flight Box at its leading sides and released at its trailing sides as the aircraft moves.

Figure 102. Rooivalk with surrounding Flight Box [A Landman, 2017]

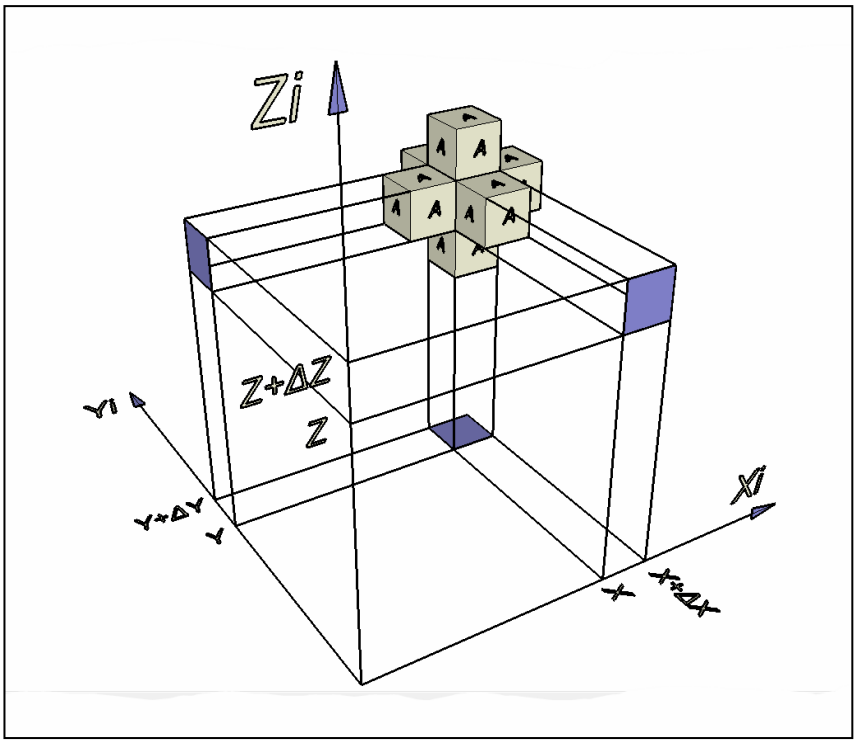


Figure 103. Adjacent neighbours of a given test volume [A Landman, 2014]

Note: *For the purposes of this thesis the occupation of test volumes by a body is referred to as* ***Commutation****. The occupation and releasing of test volumes by the Flight Box is referred to as* ***Grid Hop****.*

Note that the overall flow field around the helicopter comprises a near field (essentially the flow through and around the main rotor), the intensity of which drops off rapidly with distance to form a far field. This rapid drop-off is due to the relatively low density of air (which carries momentum) in conjunction with the relatively high viscosity of air which resists differential movement between adjacent layers of air. Since the atmosphere is subjected to earth's gravity, the downwash tends to shape itself into a dam above the ground. In such a scenario, the air removed by the main rotor from the top of the downwash causes a "hole" that must be replenished by air flowing in from the surrounding areas. Similar to an electrical circuit, this flow will follow the path of least resistance, causing the surplus air at the bottom of the main rotor to curl back to the top of the main rotor because longer path lengths present increasing resistance to flow. Hence, the disturbance of ambient air at some distance from the aircraft will become negligibly small.

The implication of this is that the downwash needs only to be solved within the Flight Box. The velocity of air at the borders of the Flight Box can be equated to the ambient wind velocity, incurring only a negligibly small error on the flow field solution.

4.5.1.1 Sign, change and naming conventions

As sign convention mass or energy added to a test volume is assumed as positive and mass or energy removed from a test volume as negative. In addition, the test volume is assumed to be located in the first quadrant (i.e. positions where x, y and z are positive) as illustrated in Figure 103 and Figure 104. No assumptions are made concerning the variables *p, T, u, v* and *w* since their signs determine whether mass or energy flow into or out of the test volume.

To express the change of a given dependant variable (e.g. *p, T, u, v* and *w*) as function of a given independent variable (e.g. *t, x, y* or *z*), such a change is defined as the value of the dependant variable at the incremented independent variable (e.g. at $t+\Delta t$ or $x+\Delta x$) minus the value of the dependant variable at the non-incremented independent variable (e.g. at *t* or *x*).

To derive a naming convention refer to Figure 68 which shows a test volume located at position [*x,y,z*] surrounded by six adjacent test volumes. For this arrangement the following is defined:

a) The pressure of the gas at position [*x,y,z*] is $P(x,y,z,t)$ at moment *t* and $P(x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
b) The pressure of the gas at position $[x+\Delta x,y,z]$ is $P(x+\Delta x,y,z,t)$ at moment *t* and $P(x+\Delta x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
c) The pressure of the gas at position $[x,y+\Delta y,z]$ is $P(x,y+\Delta y,z,t)$ at moment *t* and $P(x,y+\Delta y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
d) The pressure of the gas at position $[x,y,z+\Delta z]$ is $P(x,y,z+\Delta z,t)$ at moment *t* and $P(x,y,z+\Delta z,t+\Delta t)$ at moment $(t+\Delta t)$.
e) The velocity of the gas along the Xi axis at position [*x,y,z*] is $u(x,y,z,t)$ at moment *t* and $u(x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
f) The velocity of the gas along the Xi axis at position $[x+\Delta x,y,z]$ is $u(x+\Delta x,y,z,t)$ at moment *t* and $u(x+\Delta x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
g) The velocity of the gas along the Yi axis at position [*x,y,z*] is $v(x,y,z,t)$ at moment *t* and $v(x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
h) The velocity of the gas along the Yi axis at position $[x,y+\Delta y,z]$ is $v(x,y+\Delta y,z,t)$ at moment *t* and $v(x,y+\Delta y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
i) The velocity of the gas along the Zi axis at position [*x,y,z*] is $w(x,y,z,t)$ at moment *t* and $w(x,y,z,t+\Delta t)$ at moment $(t+\Delta t)$.
j) The velocity of the gas along the Zi axis at position $[x,y,z+\Delta z]$ is $w(x,y,z+\Delta z,t)$ at moment *t* and $w(x,y,z+\Delta z,t+\Delta t)$ at moment $(t+\Delta t)$.

The conventions as defined above must be strictly enforced in order to ensure a coherent set of equations for describing the dynamics of the Downwash Algorithm. If not, the equations result in wrong answers with instabilities in the solver that are particularly hard to debug.

4.5.1.2 Principle of operation

Assume that during algorithm start, the system is boots-trapped by setting all air velocities and partial derivatives to zero and all air pressures and air temperatures to the ambient conditions at that time. This data set serves as initial conditions for the first iteration. During subsequent iterations the output calculated in the previous iteration becomes the initial data set for the current cycle. Each iteration is started by incrementing *t* to $t+\Delta t$ which sets the new time marker.

To visualize the evolution of a typical parameter in this scheme, refer to Figure 104 which shows the evolution of a fictitious parameter *V* as function of *x* and *t*. Observe the parameter over a distance Δx and for a time period Δt with Δx and Δt small enough so that the parameter can be taken as constant within the observation interval. At $x+\Delta x$ the parameter assumes a new value and then remains constant over the distance $(x+\Delta x) \leq X < (x+2\Delta x)$. This same pattern is repeated for all the other points over the entire ***X*** axis. Similarly, the parameter is taken as constant over the period $t \leq T < (t+\Delta t)$. At $t+\Delta t$ it assumes a new value and then remains constant for the period $(t+\Delta t) \leq T < (t+2.\Delta t)$. The same pattern is repeated for other moments in time. It is evident from Figure 104 that as Δx and Δt are made increasingly smaller, the facetted surface more closely approximates the actual surface until, in the limit as Δx and Δt each approaches zero, the two surfaces become the same. This same reasoning is also applicable to

all the other dependant variables of the system along all of the axes used to describe the system.

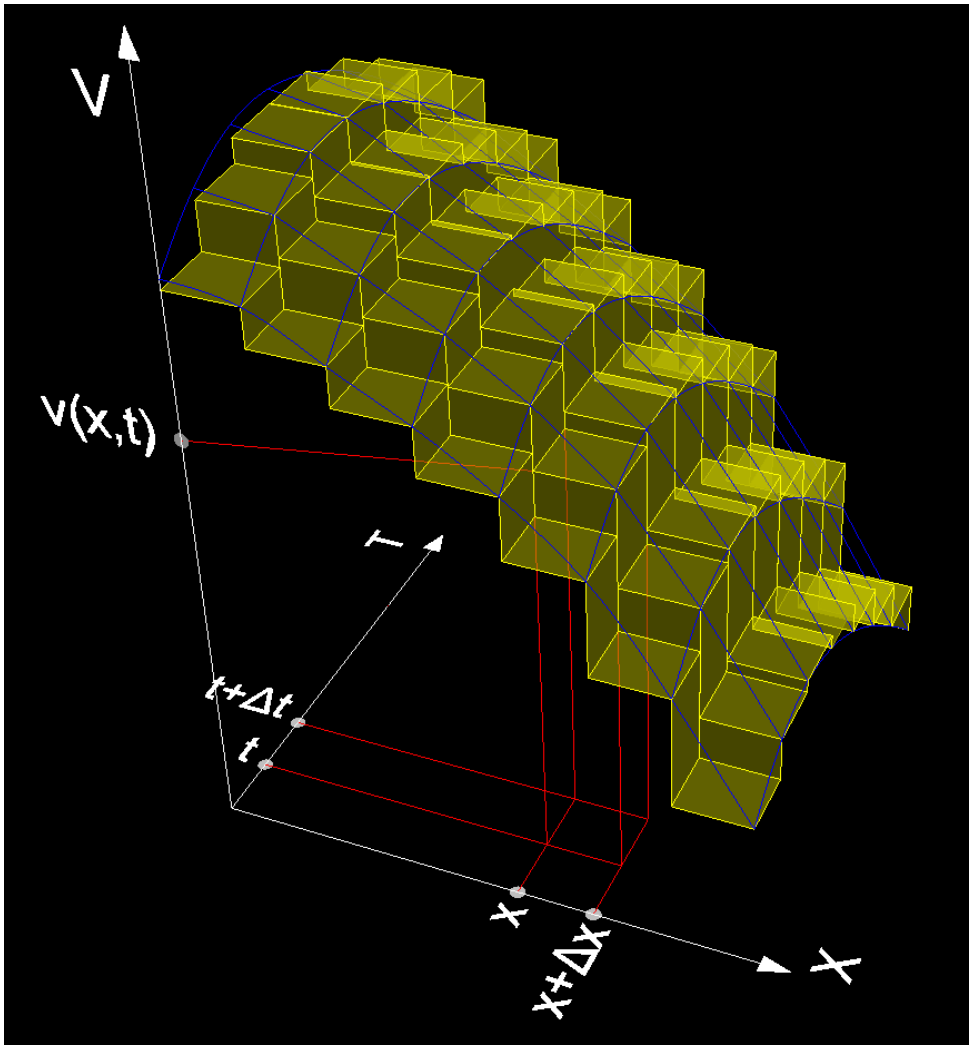


Figure 104. Facet-wise approximation for evolution of parameter V along T and X axes [A Landman, 2017]

4.5.2 Equation of state

To derive a differential equation describing the behaviour of air within the Flight Box, it is noted from observation that the temperature variation of air flowing through the main rotor is relatively small. Combined with the fact that the ambient temperature for Rooivalk AH-2A falls in the range -20°C to +50°C, it can be assumed with good certainty that the air within the downwash obeys the ideal gas law for the purposes of this study.

Under the above assumption only the temperature and pressure of the air in a given test volume need to be known to determine the density of the air at that point because the volume of the test volume is known. With temperature and pressure of the air within a given test volume known, both the mass and viscosity of the air in the test volume are effectively determined. Hence, assuming the air in the test volume to obey the ideal gas law the equation of state for the air in the test volume can now be formulated as follows:

$$\begin{aligned} pV_s &= m_s RT \\ \therefore \quad \rho_s &= \frac{p}{RT} \end{aligned}$$

Equation 74

With ρ_s the density of the air in the test volume, V_s the volume of the test volume, m_s the mass of the air in the test volume, p the pressure of the air inside the test volume, T the temperature of the air inside the test volume and R the specific gas constant for air (which has the value of 287 J/kg°K).

Note: *As a check, substituting $p=1{,}013.10^5$ N/m^2 and T=273°K into Equation 74 yields 1,2929 kg/m^3 for density which correspond to air at STP conditions.*

Note the use of the specific gas constant for air in Equation 74 and not the universal gas constant. The reason for this is that R must be expressed in MKS units because it is used in conjunction with the specific heat capacity of air which is expressd in MKS units.

The third parameter required to define the state of the air in a given test volume is air velocity (which determines air pressure and visa-versa). Air velocity is a vector with three components whereas pressure is a scalar. Hence, to solve the flow field problem five equations in temperature, pressure and velocity of the air in each test volume is required. These equations were derived from various considerations as described below.

4.5.3 Conservation of energy and mass

The basic premise for deriving the other equations required to solve for the downwash flow field is that both energy and mass must be conserved as they pass through the boundaries of each test volume. During a time interval Δt the gas in a test volume is subjected to a change in kinetic and thermal energy while work is performed on or by the test volume as the air it contains is compressed or allowed to expand. For the law of conservation of energy to hold, the sum of these changes must be zero.

It should be noted here that the Navier Stokes equations are derived on similar grounds as the approach followed here and that these equations could have been solved directly. This is the route that most authors such as *Wadcock*, *Ewing*, *Solis*, *Potsdam* and *Rajagopalan* [75] would follow. However, whereas their Computational Fluid Dynamics (CFD) simulations can run for hours on a cluster, the objective here is to do the same in real time. This is because the results produced by the Downwash Algorithm are used to estimate the trajectory of the FZ90 Rocket which in turn is used by the MPCL loop to steer the FZ90 Rocket - all in real time. Achieving real time operation in the case of the PRLS is more important than to compute an exact solution. Similar to the approach followed by *Ward* [76] the approach here was to derive a version of the Navier Stokes equation that would allow it to be tailored for speed while still achieving a solution that is good enough for the application.

Note: *For the purposes of this thesis we will refer to the equations as developed here as the Tailored Navier Stokes (TNS) equations.*

Solving a CFD problem with a digital computer normally implies using either a Finite Element Model (FEM) as was done by *Lee, Choi & Kwon* [28] or using a Finite Difference Model (FDM) as was done in the case of the Downwash Algoritm as developed here. The main motive for using a FDM is to ensure predictability which is a strong requirement for certification of such an algorithm in terms of DO-178C [20]. It also ensures that the exact same algorithm can be duplicated in a fixed number of cores with fixed interfaces that all carry the same type of data with the same formatting at the same rate.

4.5.3.1 Conservation of energy

The first energy and mass conservation relationship is based on the law of conservation of energy which must hold as energy in various forms are transformed and pass through the boundaries of the control volume. During a time interval Δt the gas in the test volume is subjected to a change in kinetic and thermal energy while work is performed on or by the air in the test volume as it is compressed or allowed to expand. For the law of conservation of energy to hold, the sum of these energy changes must be zero

To derive a relationship that describes the conservation of energy in the test volume, refer to Figure 68 which shows a test volume located at position [*x,y,z*] in the first quadrant surrounded by six adjacent test volumes. Noting that the volume of any test volume is given by $\Delta x \Delta y \Delta z$, the kinetic energy ΔE_k added to the air in the test volume during time interval Δt is given by:

$$\begin{aligned} \Delta E_k &= \left\{ \begin{array}{l} \frac{1}{2}\frac{p(x,y,z,t+\Delta t)}{RT(x,y,z,t+\Delta t)}\Delta x\Delta y\Delta z\left[u^2(x,y,z,t+\Delta t)+v^2(x,y,z,t+\Delta t)+w^2(x,y,z,t+\Delta t)\right]- \\ \frac{1}{2}\frac{p(x,y,z,t)}{RT(x,y,z,t)}\Delta x\Delta y\Delta z\left[u^2(x,y,z,t)+v^2(x,y,z,t)+w^2(x,y,z,t)\right] \end{array} \right\} \\ \Rightarrow \quad \Delta E_k &= \frac{1}{2R}\left\{ \begin{array}{l} \frac{p(x,y,z,t+\Delta t)u^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\frac{p(x,y,z,t)u^2(x,y,z,t)}{T(x,y,z,t)}+ \\ \frac{p(x,y,z,t+\Delta t)v^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\frac{p(x,y,z,t)v^2(x,y,z,t)}{T(x,y,z,t)}+ \\ \frac{p(x,y,z,t+\Delta t)w^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\frac{p(x,y,z,t)w^2(x,y,z,t)}{T(x,y,z,t)} \end{array} \right\}\Delta x\Delta y\Delta z \end{aligned}$$

Equation 75

The temperature of the air in the test volume also changes during time interval Δt because work is performed on the air inside the test volume and some of the air inside the test volume is replaced by air from the adjacent cubes that may be at different temperatures to that inside the test volume. Using the convention for the evolution of parameters with position and time, the thermal energy ΔE_t added to the gas in the test volume during time interval Δt is given by:

$$\Delta E_t + \left\{ \begin{array}{l} C_a \Delta x \Delta y\, w(x,y,z,t) \Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} T(x,y,z,t) + \\ - C_a \Delta x \Delta y\, w(x,y,z+\Delta z,t) \Delta t \dfrac{p(x,y,z+\Delta z,t)}{RT(x,y,z+\Delta z,t)} T(x,y,z+\Delta z,t) + \\ C_a \Delta y \Delta z\, u(x,y,z,t) \Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} T(x,y,z,t) + \\ - C_a \Delta y \Delta z\, u(x+\Delta x,y,z,t) \Delta t \dfrac{p(x+\Delta x,y,z,t)}{RT(x+\Delta x,y,z,t)} T(x+\Delta x,y,z,t) + \\ C_a \Delta z \Delta x\, v(x,y,z,t) \Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} T(x,y,z,t) \\ - C_a \Delta z \Delta x\, v(x,y+\Delta y,z,t) \Delta t \dfrac{p(x,y+\Delta y,z,t)}{RT(x,y+\Delta y,z,t)} T(x,y+\Delta y,z,t) + \end{array} \right\} = \left\{ \begin{array}{l} C_a \dfrac{p(x,y,z,t+\Delta t)}{RT(x,y,z,t+\Delta t)} \Delta x \Delta y \Delta z T(x,y,z,t+\Delta t) + \\ - C_a \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} \Delta x \Delta y \Delta z T(x,y,z,t) \end{array} \right\}$$

Equation 76

$$\Rightarrow \quad \Delta E_t = \frac{C_a}{R} \left\{ \begin{array}{l} \left[p(x,y,z,t+\Delta t) - p(x,y,z,t) \right] \Delta x \Delta y \Delta z + \\ \left[w(x,y,z+\Delta z,t) p(x,y,z+\Delta z,t) - w(x,y,z,t) p(x,y,z,t) \right] \Delta x \Delta y \Delta t + \\ \left[u(x+\Delta x,y,z,t) p(x+\Delta x,y,z,t) - u(x,y,z,t) p(x,y,z,t) \right] \Delta y \Delta z \Delta t + \\ \left[v(x,y+\Delta y,z,t) p(x,y+\Delta y,z,t) - v(x,y,z,t) p(x,y,z,t) \right] \Delta z \Delta x \Delta t \end{array} \right\}$$

With C_a the specific heat capacity of air (i.e. 1046,5 Joule/kg.°C at an air temperature of +25°C). Note that Equation 76 reflects the fact that the change in thermal energy of the air in the test volume over time interval Δt must be equal to the thermal energy ΔE_t added through conversion from another form of energy plus the thermal energy added as a result of mass flow into the test volume minus the thermal energy removed as a result of mass flow out of the test volume.

Due to the viscosity of air, lateral movements between the test volume and adjacent test volumes result in frictional forces acting on the air in the test volume. This means that work is performed on or by the air in the test volume, so that the frictional energy ΔE_f added to the air in the test volume during time interval Δt is given by:

$$\Delta E_f = \left\{ \begin{array}{l} \mu_a \left[\frac{u(x,y,z-\Delta z,t)-u(x,y,z,t)}{\Delta z} \right] \Delta x \Delta y \, u(x,y,z,t) + \mu_a \left[\frac{u(x,y,z+\Delta z,t)-u(x,y,z,t)}{\Delta z} \right] \Delta x \Delta y \, u(x,y,z,t) + \\ \mu_a \left[\frac{v(x,y,z-\Delta z,t)-v(x,y,z,t)}{\Delta z} \right] \Delta x \Delta y \, v(x,y,z,t) + \mu_a \left[\frac{v(x,y,z+\Delta z,t)-v(x,y,z,t)}{\Delta z} \right] \Delta x \Delta y \, v(x,y,z,t) + \\ \mu_a \left[\frac{v(x-\Delta x,y,z,t)-v(x,y,z,t)}{\Delta x} \right] \Delta y \Delta z \, v(x,y,z,t) + \mu_a \left[\frac{v(x+\Delta x,y,z,t)-v(x,y,z,t)}{\Delta x} \right] \Delta y \Delta z \, v(x,y,z,t) + \\ \mu_a \left[\frac{w(x-\Delta x,y,z,t)-w(x,y,z,t)}{\Delta x} \right] \Delta y \Delta z \, w(x,y,z,t) + \mu_a \left[\frac{w(x+\Delta x,y,z,t)-w(x,y,z,t)}{\Delta x} \right] \Delta y \Delta z \, w(x,y,z,t) + \\ \mu_a \left[\frac{w(x,y-\Delta y,z,t)-w(x,y,z,t)}{\Delta y} \right] \Delta z \Delta x \, w(x,y,z,t) + \mu_a \left[\frac{w(x,y+\Delta y,z,t)-w(x,y,z,t)}{\Delta y} \right] \Delta z \Delta x \, w(x,y,z,t) + \\ \mu_a \left[\frac{u(x,y+\Delta y,z,t)-u(x,y,z,t)}{\Delta y} \right] \Delta z \Delta x \, u(x,y,z,t) + \mu_a \left[\frac{u(x,y-\Delta y,z,t)-u(x,y,z,t)}{\Delta y} \right] \Delta z \Delta x \, u(x,y,z,t) \end{array} \right\} \Delta t$$

Equation 77

With μ_a the viscosity of air (i.e. $1{,}861.10^{-5}$ Ns/m^2 at an air temperature of +25°C). Note that μ_a changes with temperature in accordance with the Sutherland formula but that it is virtually independent of pressure. Nevertheless, the assumption here is that the temperature variation of air in the Rooivalk Rotor flow field will be small enough so that μ_a can be taken as constant.

Finally, gas is being forced into the test volume from the adjacent cubes and visa-versa. Hence, the pressure energy ΔE_p added to the air in the test volume during time interval Δt is given by:

$$\Delta E_p = \left\{ \begin{array}{l} \left[p(x,y,z+\Delta z,t) w(x,y,z+\Delta z,t) - p(x,y,z,t) w(x,y,z,t) \right] \Delta x \Delta y \\ \left[p(x+\Delta x,y,z,t) u(x+\Delta x,y,z,t) - p(x,y,z,t) u(x,y,z,t) \right] \Delta y \Delta z \\ \left[p(x,y+\Delta y,z,t) v(x,y+\Delta y,z,t) - p(x,y,z,t) v(x,y,z,t) \right] \Delta z \Delta x \end{array} \right\} \Delta t$$

Equation 78

For the law of conservation of energy to hold, the sum of Equation 75, Equation 76, Equation 77 and Equation 78 must be zero. Adding these equations and dividing throughout with $\Delta x \Delta y \Delta z$ and Δt yields:

$$\frac{1}{2R}\left\{\begin{array}{l}\dfrac{\dfrac{p(x,y,z,t+\Delta t)u^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\dfrac{p(x,y,z,t)u^2(x,y,z,t)}{T(x,y,z,t)}}{\Delta t}+\\ \dfrac{\dfrac{p(x,y,z,t+\Delta t)v^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\dfrac{p(x,y,z,t)v^2(x,y,z,t)}{T(x,y,z,t)}}{\Delta t}+\\ \dfrac{\dfrac{p(x,y,z,t+\Delta t)w^2(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\dfrac{p(x,y,z,t)w^2(x,y,z,t)}{T(x,y,z,t)}}{\Delta t}\end{array}\right\}+\frac{C_a}{R}\left\{\begin{array}{l}\dfrac{p(x,y,z,t+\Delta t)-p(x,y,z,t)}{\Delta t}+\\ \dfrac{w(x,y,z+\Delta z,t)p(x,y,z+\Delta z,t)-w(x,y,z,t)p(x,y,z,t)}{\Delta z}+\\ \dfrac{u(x+\Delta x,y,z,t)p(x+\Delta x,y,z,t)-u(x,y,z,t)p(x,y,z,t)}{\Delta x}+\\ \dfrac{v(x,y+\Delta y,z,t)p(x,y+\Delta y,z,t)-v(x,y,z,t)p(x,y,z,t)}{\Delta y}\end{array}\right\}$$

$$\mu_a\left\{\begin{array}{l}\dfrac{1}{\Delta z}\left[\dfrac{u(x,y,z+\Delta z,t)-u(x,y,z,t)}{\Delta z}-\dfrac{u(x,y,z,t)-u(x,y,z-\Delta z,t)}{\Delta z}\right]u(x,y,z,t)+\dfrac{1}{\Delta z}\left[\dfrac{v(x,y,z+\Delta z,t)-v(x,y,z,t)}{\Delta z}-\dfrac{v(x,y,z,t)-v(x,y,z-\Delta z,t)}{\Delta z}\right]v(x,y,z,t)+\\ \dfrac{1}{\Delta x}\left[\dfrac{v(x+\Delta x,y,z,t)-v(x,y,z,t)}{\Delta x}-\dfrac{v(x,y,z,t)-v(x-\Delta x,y,z,t)}{\Delta x}\right]v(x,y,z,t)+\dfrac{1}{\Delta x}\left[\dfrac{w(x+\Delta x,y,z,t)-w(x,y,z,t)}{\Delta x}-\dfrac{w(x,y,z,t)-w(x-\Delta x,y,z,t)}{\Delta x}\right]w(x,y,z,t)+\\ \dfrac{1}{\Delta y}\left[\dfrac{w(x,y+\Delta y,z,t)-w(x,y,z,t)}{\Delta y}-\dfrac{w(x,y,z,t)-w(x,y-\Delta y,z,t)}{\Delta y}\right]w(x,y,z,t)+\dfrac{1}{\Delta y}\left[\dfrac{u(x,y+\Delta y,z,t)-u(x,y,z,t)}{\Delta y}-\dfrac{u(x,y,z,t)-u(x,y-\Delta y,z,t)}{\Delta y}\right]u(x,y,z,t)\end{array}\right\}+$$

$$\left\{\begin{array}{l}\dfrac{p(x,y,z+\Delta z,t)w(x,y,z+\Delta z,t)-p(x,y,z,t)w(x,y,z,t)}{\Delta z}+\\ \dfrac{p(x+\Delta x,y,z,t)u(x+\Delta x,y,z,t)-p(x,y,z,t)u(x,y,z,t)}{\Delta x}+\\ \dfrac{p(x,y+\Delta y,z,t)v(x,y+\Delta y,z,t)-p(x,y,z,t)v(x,y,z,t)}{\Delta y}\end{array}\right\}=0$$

Equation 79

So that, in the limit as $\Delta x \to 0$ and $\Delta t \to 0$ Equation 79 reduces to:

$$\frac{1}{2R}\left\{\frac{\partial}{\partial t}\left(\frac{pu^2}{T}\right)+\frac{\partial}{\partial t}\left(\frac{pv^2}{T}\right)+\frac{\partial}{\partial t}\left(\frac{pw^2}{T}\right)\right\}+$$
$$\frac{C_a}{R}\left\{\frac{\partial p}{\partial t}+\frac{\partial}{\partial z}(pw)+\frac{\partial}{\partial x}(pu)+\frac{\partial}{\partial y}(pv)\right\}+$$
$$\mu_a\left\{v_x\frac{\partial^2 u}{\partial z^2}+v_y\frac{\partial^2 v}{\partial z^2}+v_y\frac{\partial^2 v}{\partial x^2}+v_z\frac{\partial^2 w}{\partial x^2}+v_z\frac{\partial^2 w}{\partial y^2}+v_x\frac{\partial^2 u}{\partial y^2}\right\}+$$
$$\left\{\frac{\partial}{\partial z}(pw)+\frac{\partial}{\partial x}(pu)+\frac{\partial}{\partial y}(pv)\right\}=0$$

Equation 80

Note that R, C_a and μ_a are constants while T, p, u, v and w are functions of x, y, z and t. To expand Equation 80 into a form suitable for numerical analysis, evaluate the following partial differential term:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{pu^2}{T}\right) &= \frac{\partial}{\partial t}\left(u^2\right)\frac{p}{T} + u^2\frac{\partial}{\partial t}\left(\frac{p}{T}\right)\\ &= \frac{2up}{T}\frac{\partial u}{\partial t} + u^2\left\{\frac{\partial p}{\partial t}\frac{1}{T} + p\frac{\partial}{\partial t}\left(\frac{1}{T}\right)\right\}\\ &= \frac{2up}{T}\frac{\partial u}{\partial t} + \frac{u^2}{T}\frac{\partial p}{\partial t} - \frac{p}{T^2}\frac{\partial T}{\partial t}\end{aligned}$$

Equation 81

Similarly:

$$\frac{\partial}{\partial t}\left(\frac{pv^2}{T}\right) = \frac{2vp}{T}\frac{\partial v}{\partial t} + \frac{v^2}{T}\frac{\partial p}{\partial t} - \frac{p}{T^2}\frac{\partial T}{\partial t}$$
$$\frac{\partial}{\partial t}\left(\frac{pw^2}{T}\right) = \frac{2wp}{T}\frac{\partial w}{\partial t} + \frac{w^2}{T}\frac{\partial p}{\partial t} - \frac{p}{T^2}\frac{\partial T}{\partial t}$$

Equation 82

Also:

$$\begin{aligned}\frac{\partial}{\partial z}(pw) &= p\frac{\partial w}{\partial z}+w\frac{\partial p}{\partial z}\\ \frac{\partial}{\partial x}(pu) &= p\frac{\partial u}{\partial x}+u\frac{\partial p}{\partial x}\\ \frac{\partial}{\partial y}(pv) &= p\frac{\partial v}{\partial y}+v\frac{\partial p}{\partial y}\end{aligned}$$

Equation 83

Substitution of Equation 81 thru Equation 83 into Equation 80 yields:

$$\left\{\begin{array}{l}\frac{1}{2R}\left\{\frac{2up}{T}\frac{\partial u}{\partial t}+\frac{u^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t} \quad + \quad \frac{2vp}{T}\frac{\partial v}{\partial t}+\frac{v^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t} \quad + \quad \frac{2wp}{T}\frac{\partial w}{\partial t}+\frac{w^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t}\right\}+\\ \frac{C_a}{R}\left\{\frac{\partial p}{\partial t}+w\frac{\partial p}{\partial z}+p\frac{\partial w}{\partial z}+u\frac{\partial p}{\partial x}+p\frac{\partial u}{\partial x}+v\frac{\partial p}{\partial y}+p\frac{\partial v}{\partial y}\right\}+\\ \mu_a\left\{u\frac{\partial^2 u}{\partial z^2}+v\frac{\partial^2 v}{\partial z^2}+v\frac{\partial^2 v}{\partial x^2}+w\frac{\partial^2 w}{\partial x^2}+w\frac{\partial^2 w}{\partial y^2}+u\frac{\partial^2 u}{\partial y^2}\right\}+\\ \left\{w\frac{\partial p}{\partial z}+p\frac{\partial w}{\partial z}+u\frac{\partial p}{\partial x}+p\frac{\partial u}{\partial x}+v\frac{\partial p}{\partial y}+p\frac{\partial v}{\partial y}\right\}\end{array}\right\}=0$$

Equation 84

4.5.3.2 Conservation of momentum

The second energy and mass conservation relationship is based on the requirement that the motion of the gas in the test volume must be in accordance with Newton's second law. The gas in the test volume will be subject to acceleration because the pressures acting on the faces of the gas in the test volume are different, causing the resultant forces acting on the air in the test volume to be non-zero. Hence, the force acting on the gas in the test volume along the **X**i-axis is given by:

$$\Delta F_x = -\Delta y \Delta z \left[p(x + \Delta x, y, z, t) - p(x, y, z, t) \right]$$ **Equation 85**

This force will result in acceleration of the gas in the test volume along the **X**i-axis. Combining Newton's second law and Equation 74 yields:

$$\Delta F_x = \Delta x \Delta y \Delta z \frac{M_a p(x, y, z, t)}{RT(x, y, z, t)} \frac{\partial}{\partial t} u(x, y, z, t)$$ **Equation 86**

Equating Equation 85 and Equation 86 and dividing both sides with Δx yields:

$$-\left[\frac{p(x + \Delta x, y, z, t) - p(x, y, z, t)}{\Delta x} \right] = \frac{p(x, y, y, z, t)}{RT(x, y, z, t)} \frac{\partial}{\partial t} u(x, y, z, t)$$ **Equation 87**

So that in the limit as $\Delta x \to 0$ Equation 87 reduces to the following:

$$\frac{\partial p}{\partial x} = -\frac{\partial u}{\partial t} \frac{p}{RT}$$ **Equation 88**

Similarly, for the momentums along the **Y**i-axis and **Z**i-axis we find:

$$\frac{\partial p}{\partial y} = -\frac{\partial v}{\partial t} \frac{p}{RT}$$ **Equation 89**

$$\frac{\partial p}{\partial z} = -\frac{\partial w}{\partial t} \frac{p}{RT}$$ **Equation 90**

4.5.3.3 Conservation of mass

The third energy and mass conservation relationship is based on the requirement that the change in mass of the air in the test volume must be equal to the accumulated difference in mass flow rates at the opposite faces of the control volume over time period Δt. Hence, the change in mass Δm_s of the air in the test volume can be written as follows:

$$\Delta m_s = \begin{bmatrix} -\Delta x\Delta y\, w(x,y,z+\Delta z,t)\Delta t \dfrac{p(x,y,z+\Delta z,t)}{RT(x,y,z+\Delta z,t)} + \Delta x\Delta y\, w(x,y,z,t)\Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} \\ -\Delta y\Delta z\, u(x+\Delta x,y,z,t)\Delta t \dfrac{p(x+\Delta x,y,z,t)}{RT(x+\Delta x,y,z,t)} + \Delta y\Delta z\, u(x,y,z,t)\Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} \\ -\Delta z\Delta x\, v(x,y+\Delta y,z,t)\Delta t \dfrac{p(x,y+\Delta y,z,t)}{RT(x,y+\Delta y,z,t)} + \Delta z\Delta x\, v(x,y,z,t)\Delta t \dfrac{p(x,y,z,t)}{RT(x,y,z,t)} \end{bmatrix} \quad \textbf{Equation 91}$$

This change in mass of the gas in the test volume must be reflected by associated changes in temperature and pressure of the gas in the test volume during time period Δt. Since the test volume Vs is given by $\Delta x\Delta y\Delta z$, we have from Equation 74:

$$\Delta m_s = \frac{\Delta x\Delta y\Delta z}{R}\left[\frac{p(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)} - \frac{p(x,y,z,t)}{T(x,y,z,t)}\right] \quad \textbf{Equation 92}$$

Equating Equation 91 and Equation 92 results in:

$$-\left[\begin{array}{l}\Delta x\Delta y\; w(x,y,z+\Delta z,t)\Delta t\,\dfrac{p(x,y,z+\Delta z,t)}{RT(x,y,z+\Delta z,t)}-\Delta x\Delta y\; w(x,y,z,t)\Delta t\,\dfrac{p(x,y,z,t)}{RT(x,y,z,t)}\\ \Delta y\Delta z\; u(x+\Delta x,y,z,t)\Delta t\,\dfrac{p(x+\Delta x,y,z,t)}{RT(x+\Delta x,y,z,t)}-\Delta y\Delta z\; u(x,y,z,t)\Delta t\,\dfrac{p(x,y,z,t)}{RT(x,y,z,t)}\\ \Delta z\Delta x\; v(x,y+\Delta y,z,t)\Delta t\,\dfrac{p(x,y+\Delta y,z,t)}{RT(x,y+\Delta y,z,t)}-\Delta z\Delta x\; v(x,y,z,t)\Delta t\,\dfrac{p(x,y,z,t)}{RT(x,y,z,t)}\end{array}\right]=\frac{\Delta x\Delta y\Delta z}{R}\left[\frac{p(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\frac{p(x,y,z,t)}{T(x,y,z,t)}\right]$$

$$\Rightarrow\left[\begin{array}{l}\dfrac{w(x,y,z+\Delta z,t)\dfrac{p(x,y,z+\Delta z,t)}{T(x,y,z+\Delta z,t)}-w(x,y,z,t)\dfrac{p(x,y,z,t)}{T(x,y,z,t)}}{\Delta z}+\\ \dfrac{u(x+\Delta x,y,z,t)\dfrac{p(x+\Delta x,y,z,t)}{T(x+\Delta x,y,z,t)}-u(x,y,z,t)\dfrac{p(x,y,z,t)}{T(x,y,z,t)}}{\Delta x}+\\ \dfrac{v(x,y+\Delta y,z,t)\dfrac{p(x,y+\Delta y,z,t)}{T(x,y+\Delta y,z,t)}-v(x,y,z,t)\dfrac{p(x,y,z,t)}{T(x,y,z,t)}}{\Delta y}\end{array}\right]+\left[\frac{\dfrac{p(x,y,z,t+\Delta t)}{T(x,y,z,t+\Delta t)}-\dfrac{p(x,y,z,t)}{T(x,y,z,t)}}{\Delta t}\right]=0$$

Equation 93

So that, in the limit as $\Delta x\to 0$, $\Delta y\to 0$, $\Delta z\to 0$ and $\Delta t\to 0$ Equation 93 reduces to:

$$\frac{\partial}{\partial z}\left(\frac{wp}{T}\right)+\frac{\partial}{\partial x}\left(\frac{up}{T}\right)+\frac{\partial}{\partial y}\left(\frac{vp}{T}\right)+\frac{\partial}{\partial t}\left(\frac{p}{T}\right)=0$$

Equation 94

To expand Equation 94 into the required form, we evaluate the following partial differential terms:

$$\begin{aligned}\frac{\partial}{\partial z}\left(\frac{wp}{T}\right) &= \frac{\partial w}{\partial z}\frac{p}{T} + w\frac{\partial}{\partial z}\left(\frac{p}{T}\right) \\ &= \frac{\partial w}{\partial z}\frac{p}{T} + w\left(\frac{\partial p}{\partial z}\frac{1}{T} - \frac{p}{T^2}\frac{\partial T}{\partial z}\right) \\ &= \frac{p}{T}\frac{\partial w}{\partial z} + \frac{w}{T}\frac{\partial p}{\partial z} - \frac{wp}{T^2}\frac{\partial T}{\partial z}\end{aligned}$$

Equation 95

Similarly:

$$\frac{\partial}{\partial x}\left(\frac{up}{T}\right) = \frac{p}{T}\frac{\partial u}{\partial x} + \frac{u}{T}\frac{\partial p}{\partial x} - \frac{up}{T^2}\frac{\partial T}{\partial x}$$
$$\frac{\partial}{\partial y}\left(\frac{vp}{T}\right) = \frac{p}{T}\frac{\partial v}{\partial y} + \frac{v}{T}\frac{\partial p}{\partial y} - \frac{vp}{T^2}\frac{\partial T}{\partial y}$$

Equation 96

Also:

$$\frac{\partial}{\partial t}\left(\frac{p}{T}\right) = \frac{\partial p}{\partial t}\frac{1}{T} - \frac{p}{T^2}\frac{\partial T}{\partial t}$$

Equation 97

Substituting Equation 95 to Equation 97 into Equation 94 yields the following:

$$\frac{p}{T}\frac{\partial w}{\partial z} + \frac{w}{T}\frac{\partial p}{\partial z} - \frac{wp}{T^2}\frac{\partial T}{\partial z} + \frac{p}{T}\frac{\partial u}{\partial x} + \frac{u}{T}\frac{\partial p}{\partial x} - \frac{up}{T^2}\frac{\partial T}{\partial x} + \frac{p}{T}\frac{\partial v}{\partial y} + \frac{v}{T}\frac{\partial p}{\partial y} - \frac{vp}{T^2}\frac{\partial T}{\partial y} + \frac{\partial p}{\partial t}\frac{1}{T} - \frac{p}{T^2}\frac{\partial T}{\partial t} = 0$$

Equation 98

4.5.4 Solving the combined equations

The analysis performed under paragraph 4.5.2 and 4.5.3 has yielded a set of five coupled partial differential equations in the five unknowns *p, T, u, v* and *w* which describe the dynamic behavior of the air in a single test volume. This set of equations can be summarized as shown in Equation 99 which is of course just a form of the Navier Stokes equations for a compressible fluid that complies with the ideal gas law, written in a form suitably for implementation on an Array Processor as will be shown later.

$$\left\{\begin{array}{l}\frac{M_a}{2R}\left\{\frac{2up}{T}\frac{\partial u}{\partial t}+\frac{u^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t}\quad+\quad\frac{2vp}{T}\frac{\partial v}{\partial t}+\frac{v^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t}\quad+\quad\frac{2wp}{T}\frac{\partial w}{\partial t}+\frac{w^2}{T}\frac{\partial p}{\partial t}-\frac{p}{T^2}\frac{\partial T}{\partial t}\right\}+\\ \frac{C_a M_a}{R}\left\{\frac{\partial p}{\partial t}+w\frac{\partial p}{\partial z}+p\frac{\partial w}{\partial z}+u\frac{\partial p}{\partial x}+p\frac{\partial u}{\partial x}+v\frac{\partial p}{\partial y}+p\frac{\partial v}{\partial y}\right\}+\\ \mu_a\left\{u\frac{\partial^2 u}{\partial z^2}+v\frac{\partial^2 v}{\partial z^2}+v\frac{\partial^2 v}{\partial x^2}+w\frac{\partial^2 w}{\partial x^2}+w\frac{\partial^2 w}{\partial y^2}+u\frac{\partial^2 u}{\partial y^2}\right\}+\\ \left\{w\frac{\partial p}{\partial z}+p\frac{\partial w}{\partial z}+u\frac{\partial p}{\partial x}+p\frac{\partial u}{\partial x}+v\frac{\partial p}{\partial y}+p\frac{\partial v}{\partial y}\right\}\end{array}\right\}=0$$

Equation 99

$$\frac{\partial p}{\partial x}+\frac{\partial u}{\partial t}\frac{M_a p}{RT}=0$$

$$\frac{\partial p}{\partial y}+\frac{\partial v}{\partial t}\frac{M_a p}{RT}=0$$

$$\frac{\partial p}{\partial z}+\frac{\partial w}{\partial t}\frac{M_a p}{RT}=0$$

$$\frac{p}{T}\frac{\partial w}{\partial z}+\frac{w}{T}\frac{\partial p}{\partial z}-\frac{wp}{T^2}\frac{\partial T}{\partial z}\quad+\frac{p}{T}\frac{\partial u}{\partial x}+\frac{u}{T}\frac{\partial p}{\partial x}-\frac{up}{T^2}\frac{\partial T}{\partial x}\quad+\frac{p}{T}\frac{\partial v}{\partial y}+\frac{v}{T}\frac{\partial p}{\partial y}-\frac{vp}{T^2}\frac{\partial T}{\partial y}\quad+\frac{\partial p}{\partial t}\frac{1}{T}-\frac{p}{T^2}\frac{\partial T}{\partial t}=0$$

In theory, Equation 99 should be enough to calculate an exact solution using the methods of calculus. However, it turns out that deriving an exact solution is exceptionally difficult because the equations are coupled and written in terms of the first and second partial derivatives of the unknowns. To calculate a solution we therefore have to revert to an iterative algorithm which must also be suitable for implementation on an Array Processor as described in Chapters 5, 0 and 0.

What might such an algorithm look like? After some thought on the matter it becomes evident that Equation 99 contains partial derivatives of *p,T,u, v* and *w* to both time and the spatial dimensions x, y and z. What this is actually saying is that the evolution in time of *p,T,u, v* and *w* at a given point is dependant not only on the current conditions at the point but also on the conditions in the immediate vicinity of the point. In fact, inspection of Equation 99 shows that if the solution of the flow field at such a point is known at a given point in time *t*, all of the inputs necessary for calculating the partial time derivatives of *p,T,u, v* and *w* at this point and time are available and can be calculated from Equation 99 using the normal methods of calculus. With these time derivatives known, the values of *p,T,u, v* and *w* can then be integrated up to time *t+Δt* with *Δt* chosen small enough to yield the accuracy required. It is also clear that such an algorithm will also be ideally suited to parallel processing. Formalizing this strategy yields the following algorithm:

a) Assume the solution to the flow field problem is known at discrete positions throughout the Flight Box at time *t*.
b) Assume that the positions within the Flight Box where the solution is known are spaced at increments *Δx, Δy* and *Δz.*
c) Assume that the flow field solution must be calculated at time intervals *Δt*.
d) Calculate the spatial derivatives along the ***X****i,* ***Y****i* and ***Z****i* directions of *p,T,u,v* and *w* for each position [x,y,z] within the Flight Box. To do this for a given parameter along the x-direction, use the value s(*x,y,z,t*) of the parameter at position [*x,y,z,t*] and the values s(*x-Δx,y,z,t*) and s(*x+Δx,y,z,t*) of the same parameter at the two adjacent positions [*x-Δx,y,z,t*] and [*x+Δx,y,z,t*] to set up three quadratic equations:

$$\begin{aligned} s(x-\Delta x,y,z,t) &= A(x-\Delta x)^2 + B(x-\Delta x) + C \\ s(x,y,z,t) &= A(x)^2 + B(x) + C \\ s(x+\Delta x,y,z,t) &= A(x+\Delta x)^2 + B(x+\Delta x) + C \end{aligned}$$

Equation 100

$$\Rightarrow \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} (x-\Delta x)^2 & (x-\Delta x) & 1 \\ x^2 & x & 1 \\ (x+\Delta x)^2 & (x+\Delta x) & 1 \end{bmatrix}^{-1} \begin{bmatrix} s(x-\Delta x,y,z,t) \\ s(x,y,z,t) \\ s(x+\Delta x,y,z,t) \end{bmatrix}$$

With A, B and C the coefficients of a second-order Taylor series that passes through the three data points and which is thus an approximation of the behaviour of parameter *s* along the *x*-direction around the position [*x,y,z,t*]. Hence, the estimated first and second derivatives of parameter *s* in this position can be found through repeated differentiation to *x* of Equation 100. To simplify

the calculation, employ the fact that these derivatives should be the same as those of the same Taylor series expanded around *x=0*:

$$s = A^{'}\left(x - X_o\right)^2 + B^{'}\left(x - X_o\right) + C^{'}$$

with :

$$\begin{bmatrix} A^{'} \\ B^{'} \\ C^{'} \end{bmatrix} = \begin{bmatrix} (-\Delta x)^2 & (-\Delta x) & 1 \\ 0^2 & 0 & 1 \\ (+\Delta x)^2 & (+\Delta x) & 1 \end{bmatrix}^{-1} \begin{bmatrix} s(x-\Delta x, y, z, t) \\ s(x, y, z, t) \\ s(x+\Delta x, y, z, t) \end{bmatrix} \qquad \textbf{Equation 101}$$

Where X_o represents the distance of position [x,y,z,t] from the origin along the Xi axis. Under this transformation the following follows:

$$\begin{aligned} \left.\frac{\partial s}{\partial x}\right|_{x=X_o} &\approx \left[\frac{d}{dx}\left(A^{'}\left(x - X_o\right)^2 + B^{'}\left(x - X_o\right) + C^{'}\right)\right]_{x=X_o} \\ &= \left[2A^{'}\left(x - X_o\right) + B^{'}\right]_{x=X_o} \\ &= B^{'} \end{aligned} \qquad \textbf{Equation 102}$$

Similarly:

$$\begin{aligned} \left.\frac{\partial^2 s}{\partial x^2}\right|_{x=X_o} &\approx \left[\frac{d^2}{dx^2}\left(A^{'}\left(x - X_o\right)^2 + B^{'}\left(x - X_o\right) + C^{'}\right)\right]_{x=X_o} \\ &= \left[\frac{d}{dx}\left(2A^{'}\left(x - X_o\right) + B^{'}\right)\right]_{x=X_o} \\ &= 2A^{'} \end{aligned} \qquad \textbf{Equation 103}$$

Follow the same process to derive the spatial derivatives of *p,T,u,v* and *w* along the **X**i, **Y**i and **Z**i directions. At positions that intersect objects such as the Fuselage or a Main Rotor Blade, force the values of *p,T,u,v* and *w* to correspond with a suitable boundary condition as calculated with an external model of the object (i.e. transfer data with the Commutation Process). Note that the derivates calculated here can be

estimated with greater accuracy by assuming a higher order Taylor series with suitable nodes around a given position. This approach would entail a bigger computation load and was not done for the purposes of this thesis. They can also be estimated with greater accuracy by choosing smaller values for Δx, Δy and Δz which would require a larger Array Processor. Finding a suitable balance here is an iterative process that requires multiple passes through all the levels of the design hierarchy.

e) With all the partial space derivatives known, the partial time derivatives of p,T,u,v and w can be calculated by re-arranging Equation 99 as follows:

$$\begin{Bmatrix} \frac{u^2}{2RT}+ \\ \frac{v^2}{2RT}+ \\ \frac{w^2}{2RT}+ \\ \frac{C_a}{R} \end{Bmatrix}\frac{\partial p}{\partial t} + \left\{\frac{-3p}{2RT^2}\right\}\frac{\partial T}{\partial t} + \left\{\frac{up}{RT}\right\}\frac{\partial u}{\partial t} + \left\{\frac{vp}{RT}\right\}\frac{\partial v}{\partial t} + \left\{\frac{wp}{RT}\right\}\frac{\partial w}{\partial t} = \begin{Bmatrix} -\mu_a\left\{u\frac{\partial^2 u}{\partial z^2}+u\frac{\partial^2 u}{\partial y^2}+v\frac{\partial^2 v}{\partial z^2}+v\frac{\partial^2 v}{\partial x^2}+w\frac{\partial^2 w}{\partial x^2}+w\frac{\partial^2 w}{\partial y^2}\right\}+ \\ -\left(\frac{C_a}{R}+1\right)\left(w\frac{\partial p}{\partial z}+u\frac{\partial p}{\partial x}+v\frac{\partial p}{\partial y}\right)+ \\ -p\left(\frac{C_a}{R}+1\right)\left(\frac{\partial w}{\partial z}+\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+ \end{Bmatrix}$$

$$\{0\}\frac{\partial p}{\partial t} + \{0\}\frac{\partial T}{\partial t} + \left\{\frac{p}{RT}\right\}\frac{\partial u}{\partial t} + \{0\}\frac{\partial v}{\partial t} + \{0\}\frac{\partial w}{\partial t} = \left\{-\frac{\partial p}{\partial x}\right\}$$

$$\{0\}\frac{\partial p}{\partial t} + \{0\}\frac{\partial T}{\partial t} + \{0\}\frac{\partial u}{\partial t} + \left\{\frac{p}{RT}\right\}\frac{\partial v}{\partial t} + \{0\}\frac{\partial w}{\partial t} = \left\{-\frac{\partial p}{\partial y}\right\}$$

$$\{0\}\frac{\partial p}{\partial t} + \{0\}\frac{\partial T}{\partial t} + \{0\}\frac{\partial u}{\partial t} + \{0\}\frac{\partial v}{\partial t} + \left\{\frac{p}{RT}\right\}\frac{\partial w}{\partial t} = \left\{-\frac{\partial p}{\partial z}\right\}$$

$$\left\{\frac{1}{T}\right\}\frac{\partial p}{\partial t} + \left\{\frac{-p}{T^2}\right\}\frac{\partial T}{\partial t} + \{0\}\frac{\partial u}{\partial t} + \{0\}\frac{\partial v}{\partial t} + \{0\}\frac{\partial w}{\partial t} = \begin{Bmatrix} -\frac{p}{T}\left(\frac{\partial w}{\partial z}+\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+ \\ -\frac{1}{T}\left(w\frac{\partial p}{\partial z}+u\frac{\partial p}{\partial x}+v\frac{\partial p}{\partial y}\right)+ \\ +\frac{p}{T^2}\left(w\frac{\partial T}{\partial z}+u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}\right) \end{Bmatrix}$$

Equation 104

So that, from principles of ordinary matrix algebra, the required equation is as follows:

$$\begin{bmatrix} \frac{\partial p}{\partial t} \\ \frac{\partial T}{\partial t} \\ \frac{\partial u}{\partial t} \\ \frac{\partial v}{\partial t} \\ \frac{\partial w}{\partial t} \end{bmatrix} = \begin{bmatrix} \left\{ \begin{matrix} \frac{u^2}{2RT} + \\ \frac{v^2}{2RT} + \\ \frac{w^2}{2RT} + \\ \frac{C_a}{R} \end{matrix} \right\} & \left\{ \frac{-3p}{2RT^2} \right\} & \left\{ \frac{up}{RT} \right\} & \left\{ \frac{vp}{RT} \right\} & \left\{ \frac{wp}{RT} \right\} \\ 0 & 0 & \left\{ \frac{p}{RT} \right\} & 0 & 0 \\ 0 & 0 & 0 & \left\{ \frac{p}{RT} \right\} & 0 \\ 0 & 0 & 0 & 0 & \left\{ \frac{p}{RT} \right\} \\ \left\{ \frac{1}{T} \right\} & \left\{ \frac{-p}{T^2} \right\} & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \left\{ \begin{matrix} -\mu_a \left\{ u\frac{\partial^2 u}{\partial z^2} + u\frac{\partial^2 u}{\partial y^2} + v\frac{\partial^2 v}{\partial z^2} + v\frac{\partial^2 v}{\partial x^2} + w\frac{\partial^2 w}{\partial x^2} + w\frac{\partial^2 w}{\partial y^2} \right\} + \\ -\left(\frac{C_a}{R} + 1 \right) \left(w\frac{\partial p}{\partial z} + u\frac{\partial p}{\partial x} + v\frac{\partial p}{\partial y} \right) + \\ -p\left(\frac{C_a}{R} + 1 \right) \left(\frac{\partial w}{\partial z} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + \end{matrix} \right\} \\ \left\{ -\frac{\partial p}{\partial x} \right\} \\ \left\{ -\frac{\partial p}{\partial y} \right\} \\ \left\{ -\frac{\partial p}{\partial z} \right\} \\ \left\{ \begin{matrix} -\frac{p}{T}\left(\frac{\partial w}{\partial z} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + \\ -\frac{1}{T}\left(w\frac{\partial p}{\partial z} + u\frac{\partial p}{\partial x} + v\frac{\partial p}{\partial y} \right) + \\ +\frac{p}{T^2}\left(w\frac{\partial T}{\partial z} + u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} \right) \end{matrix} \right\} \end{bmatrix}$$

Equation 105

f) Multiply the time derivatives of *p,T,u,v* and *w* with the time interval *Δt* and add the results to the current values of *p,T,u,v* and *w* to yield the values for these unknowns at position [x,y,z,*t+Δt*].

g) Repeat steps (d), (e) and (f) for all the test volumes within the Flight Box to yield the complete flow field estimate at time *t+Δt*. This yields the surface as described in paragraph 4.5.1.2.

4.5.5 Ensuring stability of the solver

Given Equation 105 for estimating the state of a test volume, suitable values for Δt, Δx, Δy and Δz that are applicable to Rooivalk AH-2A now have to be chosen. Since the solver as implied by Equation 105 is discrete (i.e. sampled), the criterium as formulated by *Nyquist* [19] is applicable in both the temporal and spatial domains.

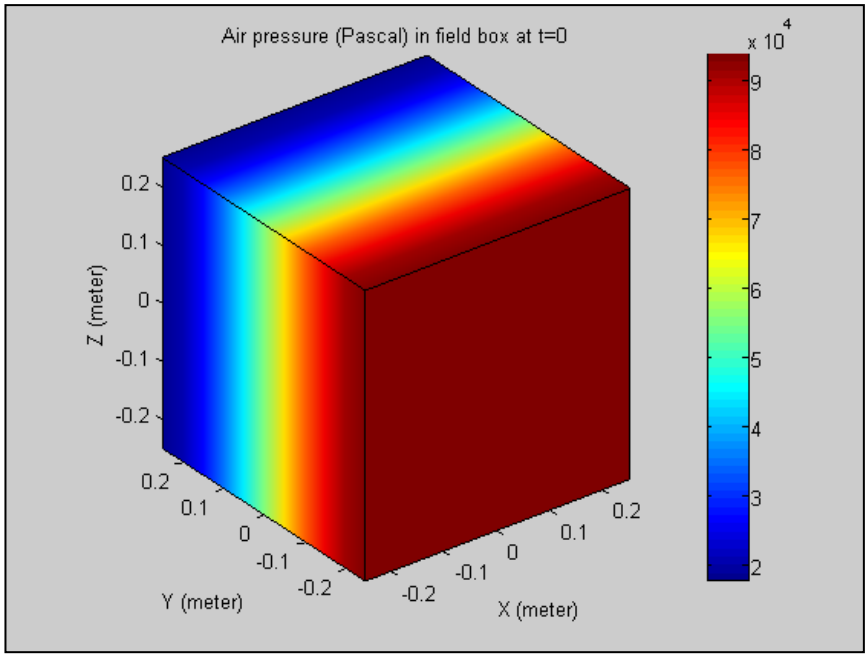


Figure 105. Initial pressure distribution of Small Box experiment [A Landman, 2014]

To evaluate the effects of using different values for Δt, Δx, Δy and Δz in Equation 105, a Matlab program[9] was written in which Equation 105 and solver as described in paragraph 4.5.4 was duplicated in a 24x24x24 mesh to model a 500x500x500 mm box of air surrounded by an adiabatic boundary. Initial conditions at t = 0 were chosen with air temperature throughout the box at 273°K and the -Y side of the box at a pressure of 1,013.10+5 Pascal (i.e. 1 atmosphere) which gradually diminished to a tenth of this pressure on the +Y side of the box. This simple "Small Box" scenario was chosen because the first-order response of such a "drumming" system can be estimated with relative ease, allowing general operation of the model to be verified through inspection.

The amount of data generated by the Small Box experiment is huge and to visually illustrate the result is not an easy task because the data represents the time evolution of five volumetric data fields. At best, some well-chosen sub-sets of the data can be displayed in an attempt to gain some understanding of what is going on. For the purpose of this thesis the spatial response of a slice of the gas in the small box over the XY plane at Z=0 and the temporal response of the gas at the center of the small box were investigated in detail.

[9] See Matlab script NSbasic.m in Appendix C

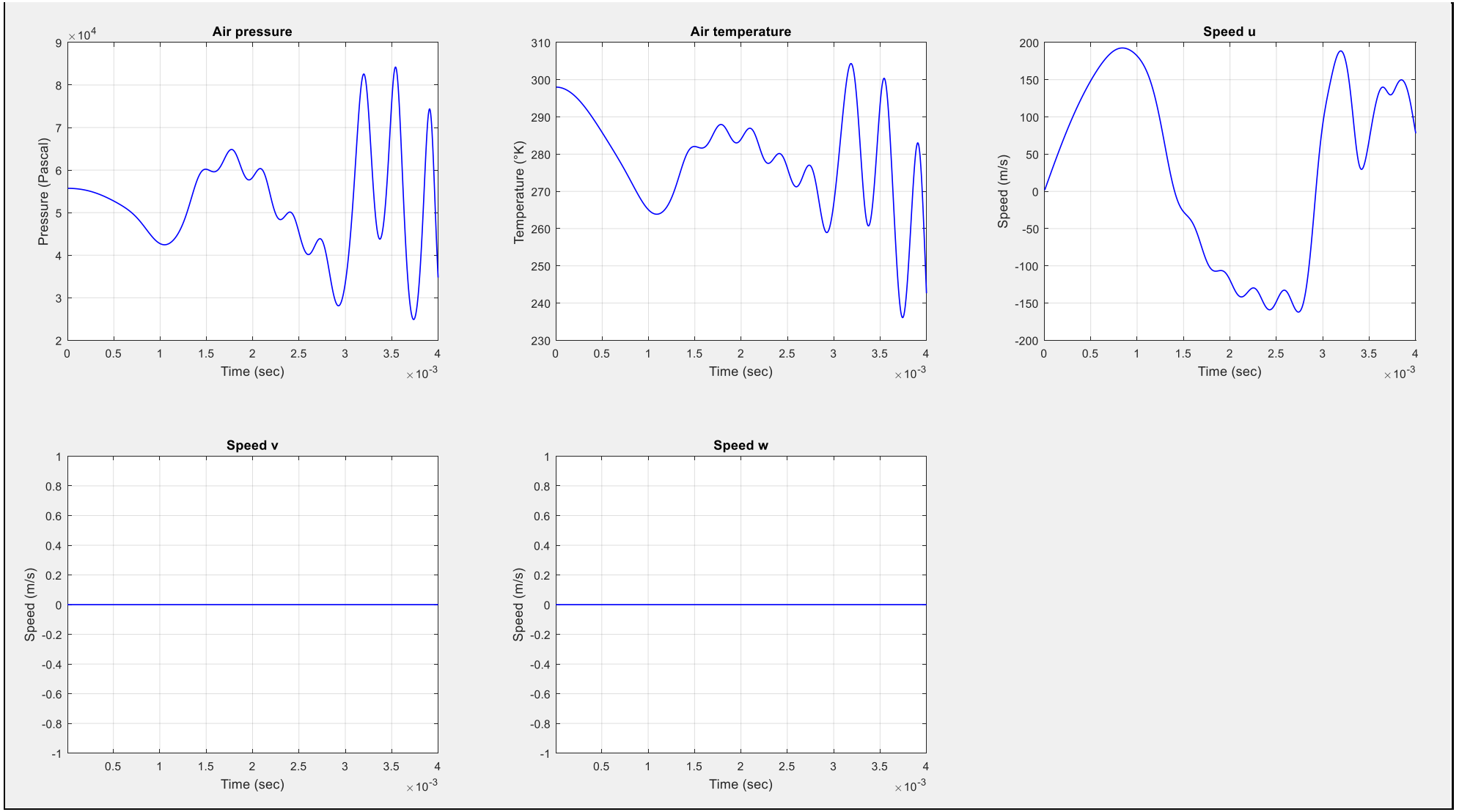


Figure 106. Time evolution of conditions at Small Box centre: mesh=20mm, Δt=5μs [A Landman, 2017]

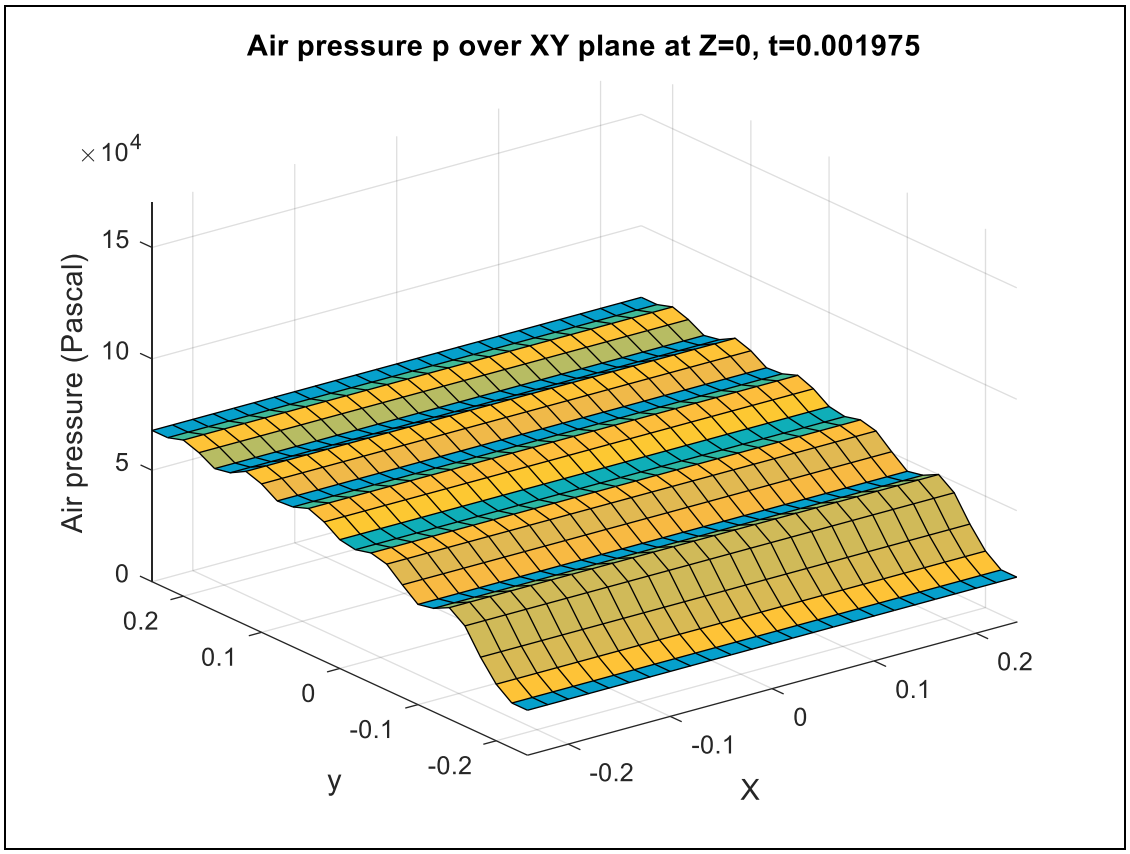


Figure 107. Spatial evolution of conditions over XY plane at Z=0: mesh=20mm, Δt=5μs [A Landman, 2017]

The spatial response of a slice of the gas over the XY plane at Z=0 as simulated with script *NSbasic.m* for a 500x500x500mm box, Δt=5μS and $\Delta x=\Delta y=\Delta z=mesh$=20mm is illustrated in Figure 107. The temporal response of the test volume at centre of the box during the same experiment is illustrated Figure 106. From these and other similar simulation runs it is evident that the solution for the Taylored Navier Stokes equation as described by Equation 105 comprises a set of sinusoidal components in both the temporal and spatial domains with frequencies ranging from zero to infinity and with amplitudes initially zero and increasing exponentially with time. The time constants of these components appear to be inversely proportional to frequency (i.e. lower frequency components grow faster in amplitude).

As partial proof of the validity of the model as developed thus far we note from Figure 106 that the first frequency component of the estimated solution has a time period of about 1,75 msec (i.e. about 570 Hz). This is the basic "drumming" mode of the small box which is caused by the air accelerating from the high-pressure wall towards the low-pressure wall where it is reflected to repeat the whole process. The simulation therefore predicts that it takes the wavefront 1,73 msec to travel a distance of 500mm. This implies a speed of 285 m/s or Mach 0,86 which is a reasonable result.

Note: *The value of zero for the velocity components v and w as illustrated in Figure 106 is correct because the test volume at the centre of the test box happens to be located on a standing wave node.*

When viewing a video comprising sequential images in time as produced by the Small Box experiment it becomes evident that as the "spatial ripple" as illustrated in Figure 107 grows in amplitude, new ripples with shorter wavelengths appear with similar evolution path albeit slower. Eventually, these ripples grow so large in amplitude that they cause the program to crash. The results of many of these simulations using different mesh and time steps strongly suggest that

the conclusion is also applicable in the limit where *mesh* and Δt are made infinitely small. In other words, the solution calculated here is a sampled version of a general solution. It is evident that Equation 105 represents a so-called "stiff" differential equation (i.e. it has a solution that contains a large number of small-amplitude terms with frequencies well above the frequency range as required by this study).

Since the approach followed here in solving Equation 105 is a discrete one using finite values for *mesh* and Δt the limitations of the Nyquist criteria [19] must be at work. Spatial frequencies in excess of *0,5/mesh* (i.e. in units of samples/m) and temporal frequencies in excess of $0,5/\Delta t$ (i.e. in units of Hz) can therefore not be represented without error using the model developed thus far. In particular, since the solution to the Navier Stokes equation actually contains components with frequencies up to infinity, the algorithm as developed here must be modified in such a way that components with high-frequencies in excess of Nyquist are filtered out and prevented from evolving in amplitude. If not, they result in aliasing and an erroneous solution as observed.

Note that removal of high-frequency components is quite acceptable here, since the requirement is to determine the Navier Stokes solution with spatial resolution an order of magnitude finer than the physical size of the Rocket and temporal resolution an order of magnitude finer than the typical response time of the Rocket (the latter requirement has a caveat as discussed later).

Note: *The algorithm used for calculating Figure 106 and Figure 107 did not include anti-aliasing filters. It only worked because the amplitudes of the high-frequency components were still small during the first 3 msec of the experiment. As will be shown later, these components causes the unprotected algorithm to crash when the simulation extends over a long enough time.*

It should be noted here that frequency components similar to those along the **X**i axis also develop along the **Y**i and **Z**i axes in the Small Box Experiment, albeit over a longer period of time. These components seem to be induced by the viscosity of the air, causing boundary layers to develop close to the walls of the test box (we know this because these components do not appear when viscosity of air is chosen as zero in the simulation). The boundary layers in turn generate pressure gradients close to the walls of the box which cause movements of the air normal to the wall surfaces that eventually leads to the cacophony of frequency components along all three of the axes as revealed by the simulation.

It is evident that with time, the air at each point within the Small Box develop a rich range of cyclical velocity components along all three of the axes. Since these velocity components occur simultaneously, they resemble vortex movements of the air which extends in size from the basic dimensions of the box to extremely small (atomic scales). It is probably true to say that the Small Box Experiment is a demonstration of the increase in entropy of the air in the test box as the initial pressure energy in the air is converted into thermal energy. In other words, the air in the test box would finally settle with zero average movement and a homenous pressure throughout the test box but with air atoms vibrating more (i.e. the air will be at higher temperature).

4.5.6 Improving the temporal estimator

In order to compare the effect of different time step sizes on the solution as generated by the solver of paragraph 4.5.4, the temporal response of the air in the test volume at the center of a Big Box with dimensions 100x100x100 meter was calculated for a mesh size of 50mm and various Δt. The result is illustrated in Figure 108, showing that the solution diverges more rapidly with larger time steps. At first sight the result for a time step of 10µsec is not expected, since at that rate the system should be able to handle temporal signals up to 50kHz which is way above the 1300Hz as observed. However, some thought on the matter shows that this behaviour is typical of the growth in error when solving a differential equation with a low order

estimator. Various schemes of higer order such as the Runge Kutta algorithm were invented to avoid this type of error.

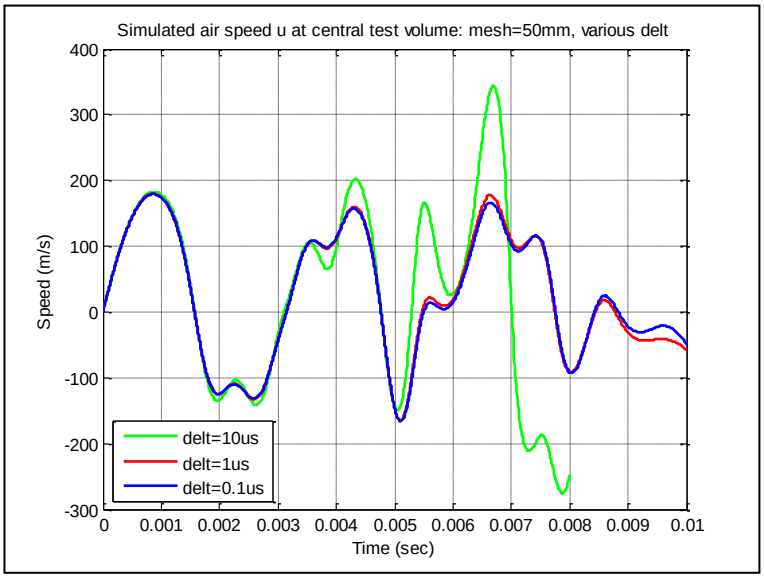


Figure 108. Growth in error of first-order solver output at various time step sizes [A Landman, 2017]

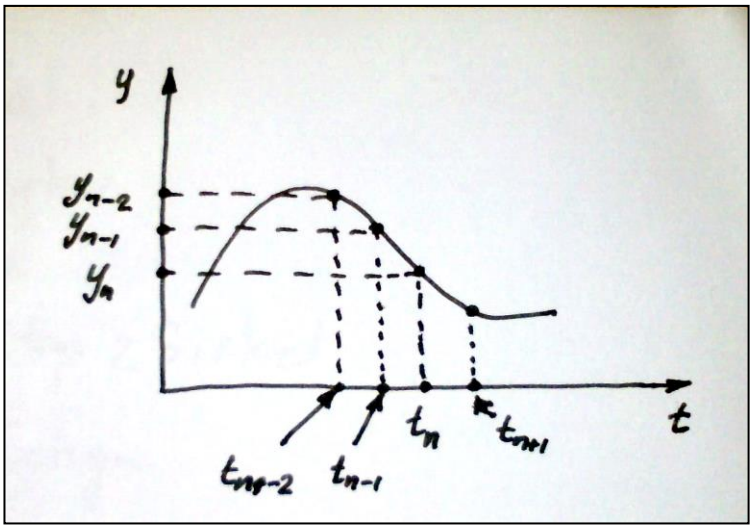


Figure 109. Taylor Series fitted to output of solver [A Landman, 2017]

To understand the mechanism for the observed growth in error, note that the solver as developed in paragraph 4.5.4 is actually a first-order solver for estimating the new values of *p, T, u, v* and *w* given their current values and their partial derivatives as calculated with Equation 105. This mechanism must result in some small error during each time step of the simulation which in turn, must result in a slight deformation of the estimated spatial fields which is again used to calculate a new set of temporal derivatives with the use of Equation 105. Hence, there must be at work here the integration of a small error multiplied by some or other loop gain that may well be very large. This same type of instability was also found by *Foster* and *Metaxas* [73] and solved by *Stam* using his “Stable Fluids” method [69].

The accuracy of the solver as developed in paragraph 4.5.4 to estimate the value at time t_{n+1} can be improved by also considering previous values of a given parameter in addition to the current value. To do this, consider Figure 109 in which a second-order Taylor series has been fitted to

the current and previous values of parameter *y* as estimated by the solver. For such a situation the equation of the Taylor series must be:

$$\begin{aligned} y &= A_2(t-t_n)^2 + B_2(t-t_n) + C_2 \\ &= A_2\Delta t^2 + B_2\Delta t + C_2 \quad with\ \Delta t = (t-t_n) \end{aligned}$$ **Equation 106**

With B_2 the slope of the curve at the current time t_n as given by Equation 105 and A_2 and C_2 constants that must be determined under the assumption that the curve passes through the previous two values y_{n-1} and y_n as illustrated in Figure 109. With constants A_2, B_2 and C_2 known, an estimate of output y_{n+1} at time t_{n+1} can then be produced. First, setting $t=t_n$ yields:

$$C_2 = y_n$$ **Equation 107**

Also:

$$\begin{aligned} y_{n-1} &= A_2(-\Delta t)^2 + B_2(-\Delta t) + C_2 \\ \Rightarrow\ B_2 &= \frac{y_{n-1} + B_2\Delta t - C_2}{\Delta t^2} \end{aligned}$$ **Equation 108**

Inserting Equation 107 and Equation 108 into Equation 106 yields the required second-order temporal estimator:

$$\begin{aligned} y_{n+1} &= y_{n-1} + B_2\Delta t - y_n + B_2\Delta t + y_n \\ &= y_{n-1} + 2B_2\Delta t \end{aligned}$$ **Equation 109**

Note that Equation 109 now replaces step (f) of the solver as developed in paragraph 4.5.4. Conducting the Small Box Experiment with this new solver proved very promising barring other effects as described below. Note that similar to the approach followed here the Stable Fluid method of *Stam* [69] involves tracing the current state variables back to their values a time Δt ago and using these values to derive a new estimate based on extrapolation and summation between adjacent test volumes. From a control perspective the *Stam* method seems to represent a lead compensator operating from the past, introducing a phase shift such as to render the algorithm stable in current time. In the case of this study the Taylor series performs a similar function by estimating the value of $\mathrm{X_{n+1}}$ based on $\frac{\partial \mathrm{X}}{\partial \mathrm{t}}$ and the value of $\mathrm{X_{n-1}}$. In other words, a moderation of the value of $\frac{\partial \mathrm{X}}{\partial \mathrm{t}}$ based on history similar to the method of *Stam*.

An attempt to improve the performance of the solver even more by using a third-order Taylor series proved unsuccessful. The reason for this is that a third-order Taylor series behaves too wildly beyond the limits set by the known vertices as indicated in Figure 109. This causes the estimated value of variable *y* at time t_{n+1} to have excessive error which renders the algorithm completely unstable.

4.5.7 Choosing a temporal anti-aliasing filter

The performance of the second-order temporal estimator running with a time step of 0,1 µS versus the original temporal estimator running with a time step of 1 µS is illustrated in Figure 110. It is clear that the second-order temporal estimator: has good accuracy at larger time steps but that it is bedevilled by the onset of instability after about 5 msec which grows in amplitude as before.

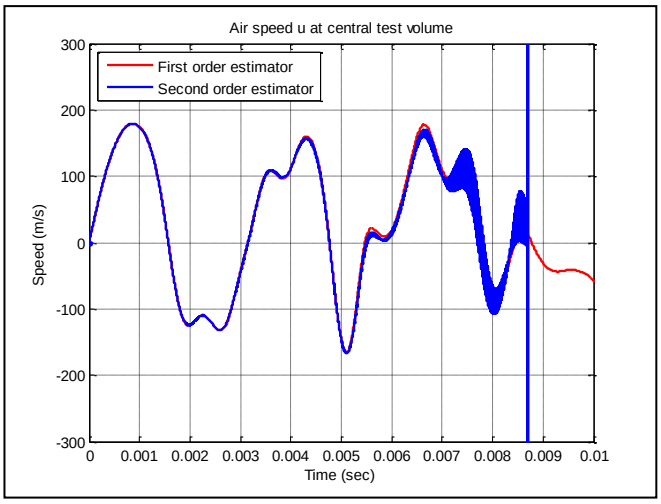


Figure 110. Instability of second-order temporal estimator [A Landman, 2014]

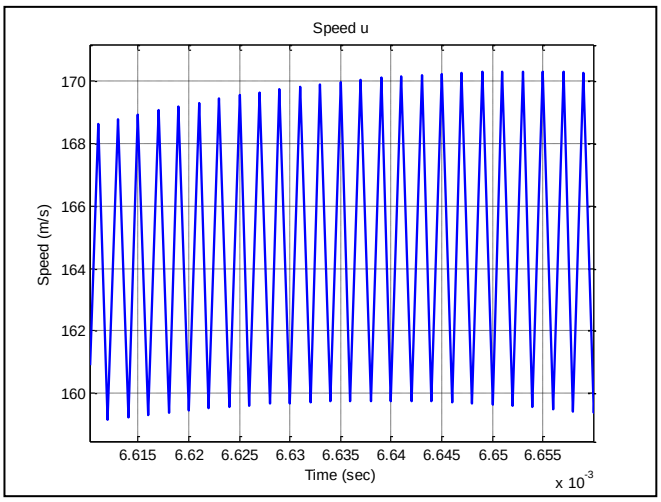


Figure 111. Alias oscillation in second-order temporal estimator output [A Landman, 2014]

Zooming into the oscillation as observed in Figure 110 reveals the pattern of Figure 111 which shows that the instability is an oscillation at 500 kHz. This is an example of a high-frequency alias in the solution that cannot be removed with Equation 109. To solve the problem, the output of the temporal estimator should also be filtered in the time domain to remove any high frequency terms that may still persist. To do this a a simple moving average filter that can readily be implemented on the Array Processor was chosen as temporal anti-aliasing filter:

$$h'_{x,y,z,t} = \frac{\left(CT_1 y_{x,y,z,t-2\Delta t} + CT_2 y_{x,y,z,t-\Delta t} + CT_3 y_{x,y,x,t}\right)}{CT_1 + CT_2 + CT_3} \qquad \textbf{Equation 110}$$

With $h'_{x,y,z,t}$ the time-filtered version of signal y at point [x,y,x,t] and CT_1, CT_2 and CT_3 a set of coefficients that must be chosen to yield the required frequency response in the time domain.

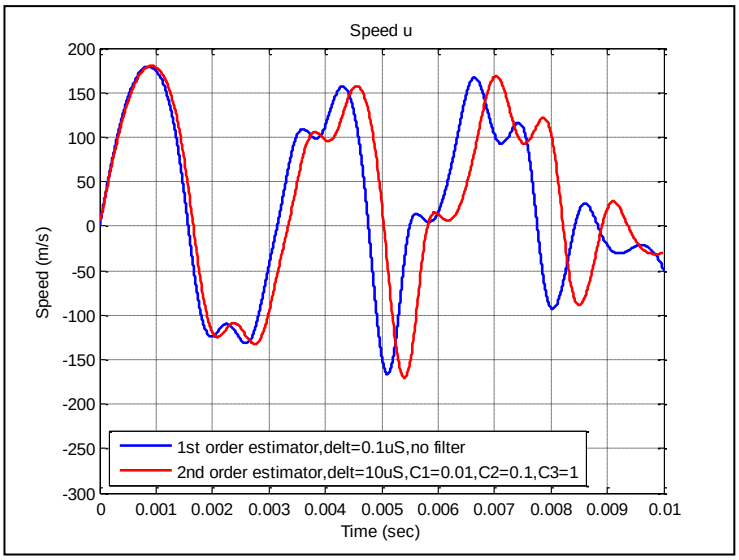


Figure 112. Second-order estimator performance with Temporal Filter1 [A Landman, 2014]

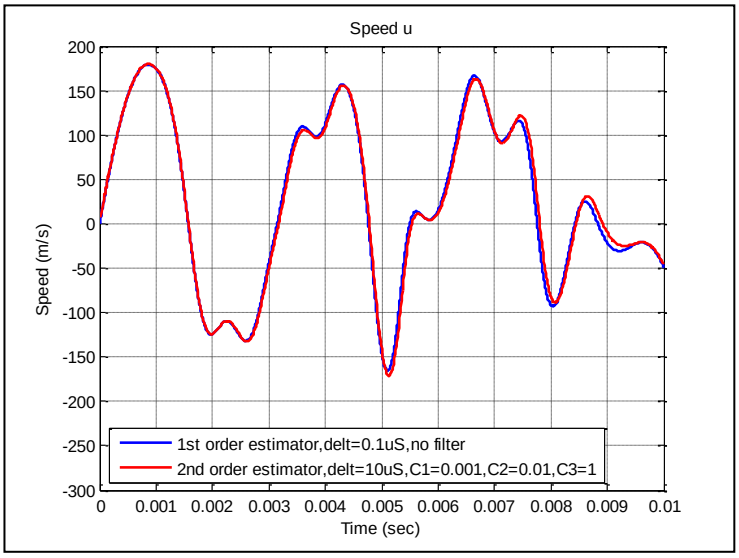


Figure 113. Second-order estimator performance with Temporal Filter2 [A Landman, 2014]

The performance of the filtered second order temporal estimator was evaluated by performing the Small Box experiment using two guessed sets of Temporal Filter Coefficients was calculated with a Matlab script and is illustrated in Figure 112 and Figure 113. It is evident that the scaling of the temporal anti-aliasing filter is not important since both filters were able to stop the instability of the algorithm without affecting the actual shape of the solution, even though the time step was two orders of magnitude larger than the reference in both cases.

Since the slower temporal anti-aliasing filter introduces a larger time shift, the filter was chosen just fast enough to pass the required temporal signals of the Taylored Navier Stokes solution. Some experimentation showed that a reasonable solution is achieved by choosing a suitably small time step with C_1=0 and C_2=C_3=0,5.

It should be noted here that as implemented here, Equation 110 actually represents a type of recursive filter, since a given output parameter at any point in time is dependant on the same parameter that has already been filtered with the same filter during previous time steps as

described by *Steiglitz* [77]. This is actually a good characteristic, since any evolving high-frequency component will continually be washed out of the solution.

4.5.8 Choosing a spatial anti-aliasing filter

Similar to the temporal domain the second-order temporal estimator developed thus far is also subject to aliases above Nyquist in the spatial domain. These spatial aliases were removed by applying a 3-dimensional moving average spatial filter that would also be easy to implement on the Processor Array:

$$h'_{x,y,z,t} = \frac{\left\{\begin{array}{l} CS_1 h_{x,y,z,t} + \\ CS_2\left(h_{x-\Delta x,y,z,t} + h_{x+\Delta x,y,z,t}\right)+ \\ CS_3\left(h_{x,y-\Delta y,z,t} + h_{x,y+\Delta y,z,t}\right)+ \\ CS_4\left(h_{x,y,z-\Delta z,t} + h_{x,y,z+\Delta z,t}\right) \end{array}\right\}}{CS_1 + 2CS_2 + 2CS_3 + 2CS_4}$$

Equation 111

With $h'_{x,y,z}$ the spatially filtered version of variable h at point $[x,y,z,t]$ and CS_1, CS_2, CS_3 and CS_4 a set of coefficients that must be chosen to yield the required frequency response in the spatial domain.

In order to evaluate the effect of such a spatial anti-aliasing filter on the Taylored Navier Stokes solution, a “Big Box” experiment similar to the Small Box experiment was arranged in which the size of the test box was enlarged to 100m x 100m x 100m (i.e. roughly an order of magnitude larger than the rotor diameter of Rooivalk AH-2A). This arrangement was chosen in order to evaluate the performance of the solver with the temporal and spatial filters of Equation 110 and Equation 111 included, given a system with temporal and spatial wavelengths similar to those of the Rooivalk AH-2A downwash flow field.

Some experimentation showed that choosing CS_1=1,0 with CS_2=CS_3=CS_4=0,01 results in a stable solver with good temporal and spatial response. The solution of the Big Box experiment as derived with these coefficients, a mesh size of 2 meter and a time step of 1msec is illustrated in Figure 114, illustrating that such a solver would be suitable for simulating the downwash of Rooivalk AH-2A.

Note: *Cognisance should be taken here of the fact that the objective here is not to find a perfect solution but a solution that is both fast and (just) good enough to achieve the PRLS accuracy as stated in paragraph 1.4. Finding a perfect estimate of the downwash flow field is not the objective here.*

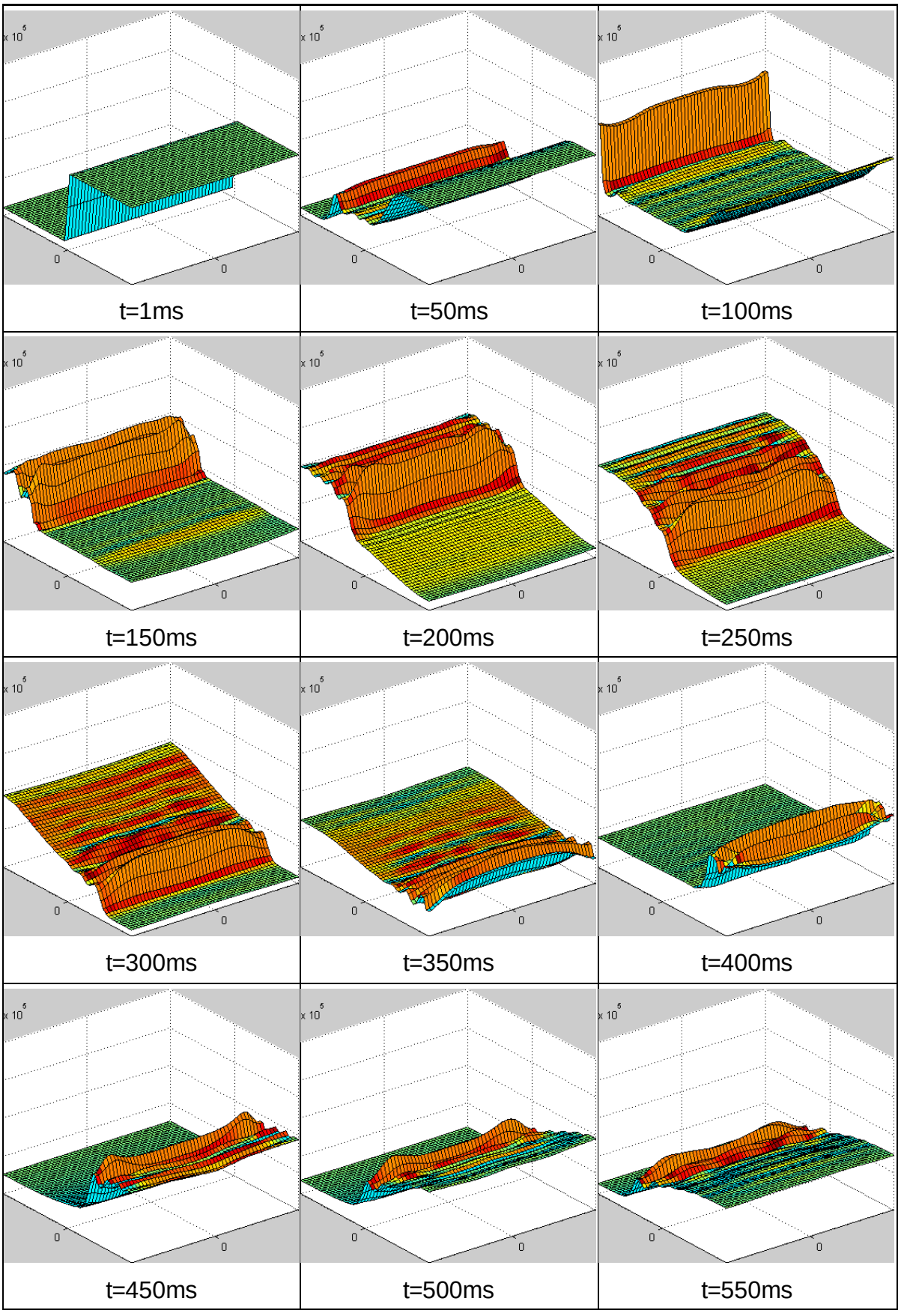


Figure 114. Evolution of pressure over XY plane at Z=0 in Big Box experiment [A Landman, 2014]

4.5.9 Testing the Donut Predictions

To determine the basic downwash pattern (i.e. downwash generated by only the main rotor without any constrictions) of Rooivalk AH-2A the solver as described in paragraphs 4.5.4 thru 4.5.8 was coded in a MATLAB program[10] referred to as the **Sequential Downwash Model** for the purposes of this study. This model assumes a Flight Box of 100x100x100 meter in dimension.

Note: *The* **Sequential Downwash Model** *is a preliminary implementation of the downwash algorithm which runs in a sequential fashion on a Von Nieuwmann machine. In this form it is relatively slow, requiring several minutes (on a laptop with 2,2 GHz clock) to perform the simulation as illustrated in Figure 114. In the final implementation (Array Processor) as developed in Chapters 5, 0 and 0 the individual parts of the algorithm run in parallel to achieve real time performance.*

Note that the **Downwash Model** forms a sub-system of the **Space** block of Figure 20 since space also serves as medium for other phenomena such as electromagnetic radiation. To simulate the basic downwash pattern of Rooivalk AH-2A the **Sequential Downwash Model** was stimulated with the rotor disk flow as calculated with the **Main Rotor** block as described in Chapter 3. Some of the results of this simulation that are useful for describing the behaviour of the downwash flow field are illustrated in Figure 115, Figure 116, Figure 117, Figure 118, Figure 119 and Figure 120.

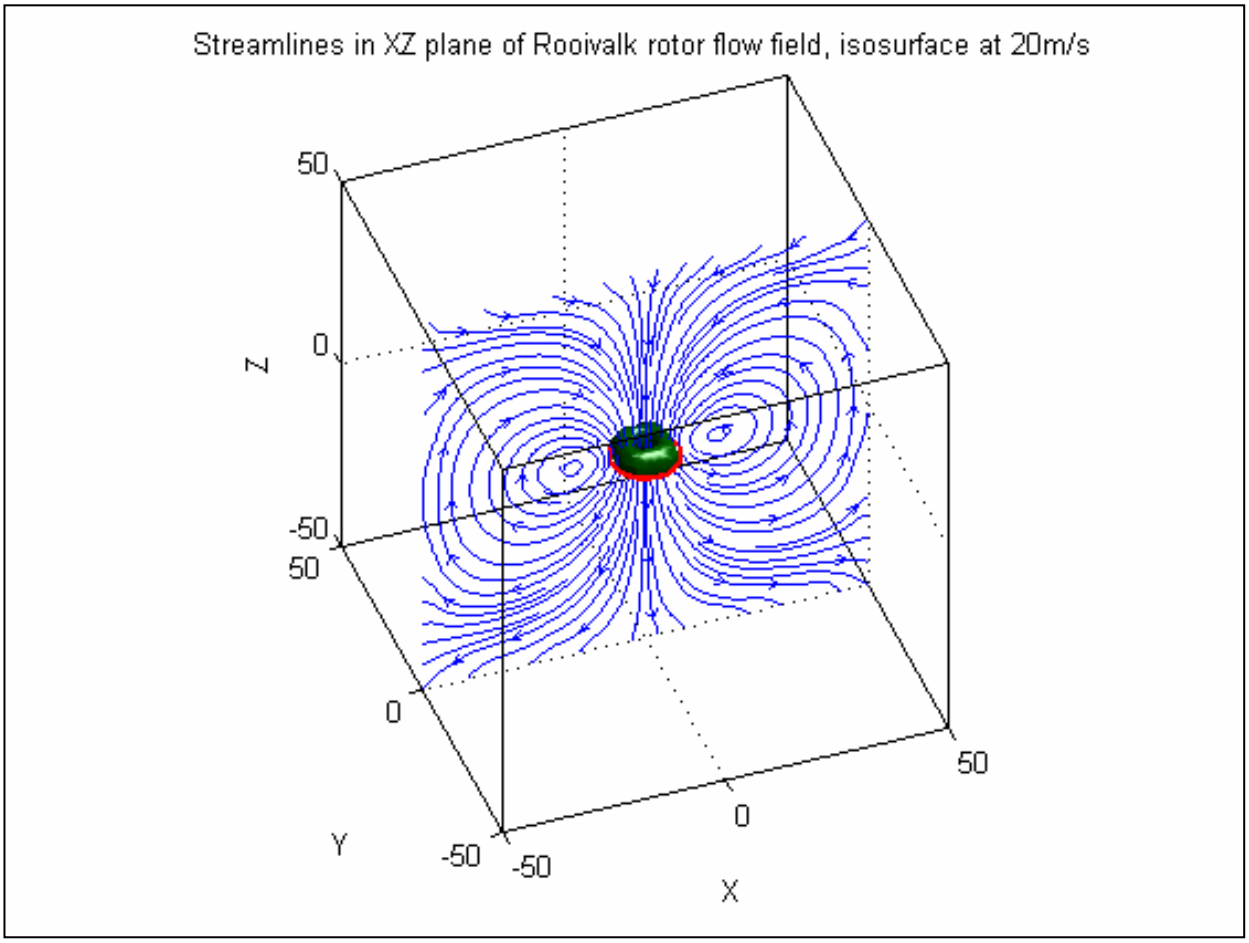


Figure 115. Streamlines at y=0 plane: Rooivalk AH-2A hover at 50m [A Landman, 2014]

[10] Refer Appendix C

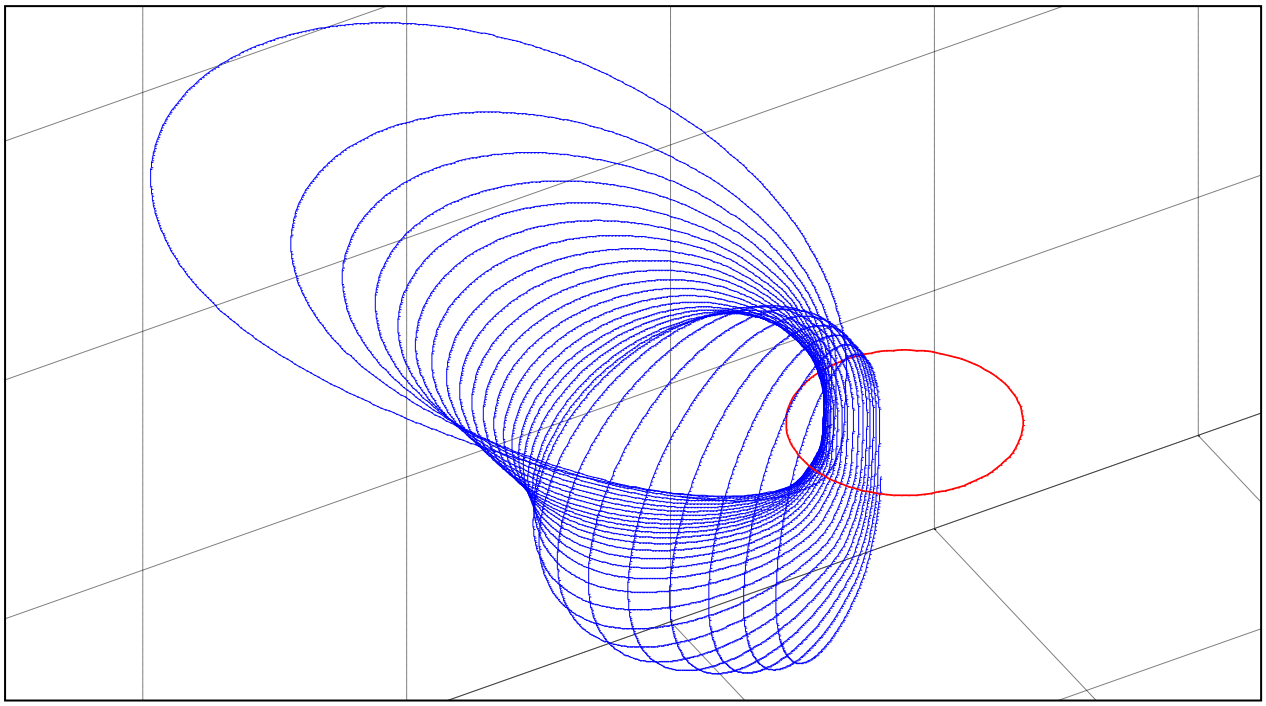


Figure 116. Path of a small particle released near rotor edge of Rooivalk AH-2A [A Landman, 2014]

Figure 115 illustrates the streamlines of the downwash flow pattern within the ZX plane at y=0. It is evident that the flow pattern as predicted by the **Sequential Downwash Model** is in agreement with the downwash patterns as observed in Figure 97, Figure 98 and Figure 99. Figure 116 illustrates the path of a small particle released from the edge of the main rotor disk. It is evident that the path of the particle follows the shape of a donut but that it also traces a helix as it is dragged along by the vertical velocity component of the downwash flow field (created by the drag of the main rotor blades).

The iso-surfaces of the downwash flow field in the absence of ambient wind within the ZX plane at y=0 are illustrated in Figure 117 and Figure 118. It is evident that the maximum velocity of the downwash flow field of Rooivalk AH-2A during both Out Of Ground (OGE) and In Ground Effect (IGE) conditions in the absence of ambient wind is about 22 m/s (versus 35,7 m/s as originally estimated in paragraph 2.5.8.4). Comparison of Figure 117 and Figure 118 shows that the ground has a prominent effect on the shape of the downwash flow pattern. In other words, the trajectory of the FZ90 Rocket depends on how close Rooivalk AH-2A is flown above ground.

It is evident from Figure 117 and Figure 118 that in the absence of ambient wind the downwash wind speed falls below 1 m/s at a distance of 40 meters from the centre of the Main Rotor. If 1,0 m/s is assumed as negligibly small for disturbing the trajectory of the FZ90 Rocket then a Flight Box of 100x100x100 meter should be adequate for calculating the downwash flow field of Rooivalk AH-2A in the absence of ambient wind.

The iso-surfaces of the downwash flow field in the presense of 10 m/s ambient wind within the ZX plane at y=0 are illustrated in Figure 119 and Figure 120. It is evident that in both cases the downwash wind speed has fallen to within 1,0 m/s of the ambient wind speed of 10 m/s. Hence, a Field Box of 100x100x100 meter should also be adequate for calculating the downwash flow field of Rooivalk AH-2A in the presence of ambient wind up to 10 m/s.

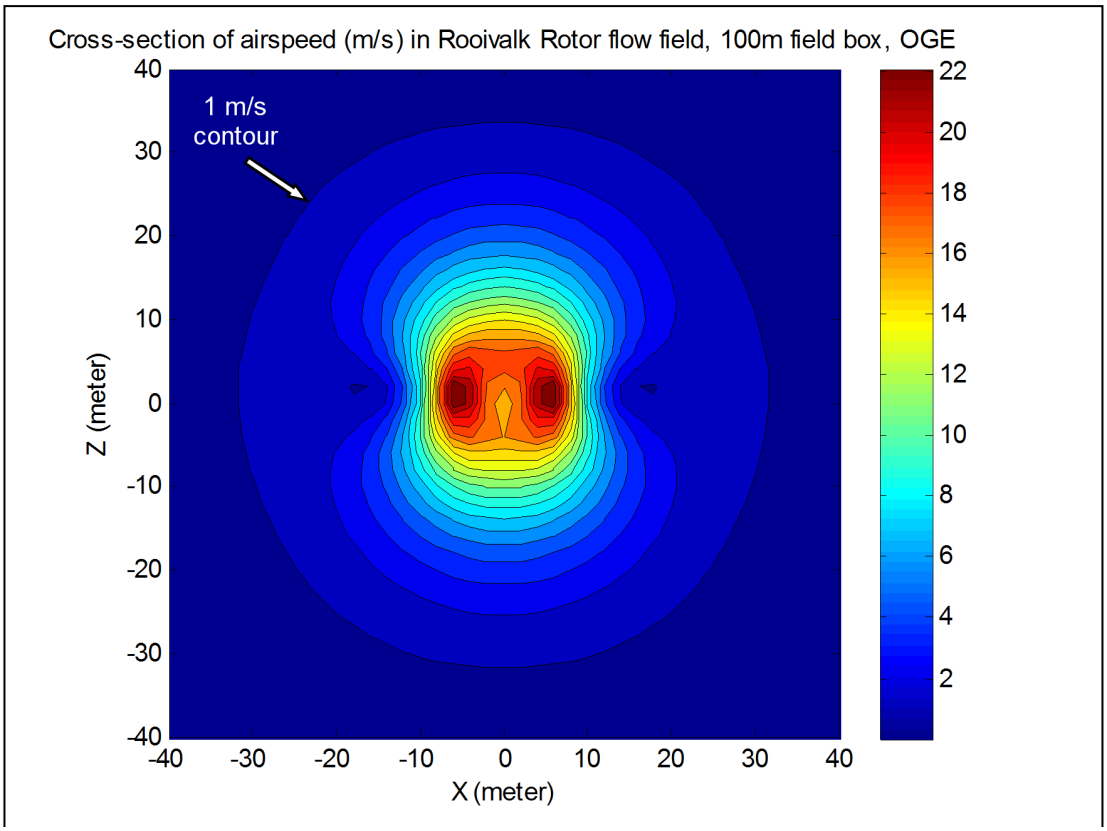


Figure 117. Rooivalk downwash: OGE hover, 100m Flight Box [A landman, 2014]

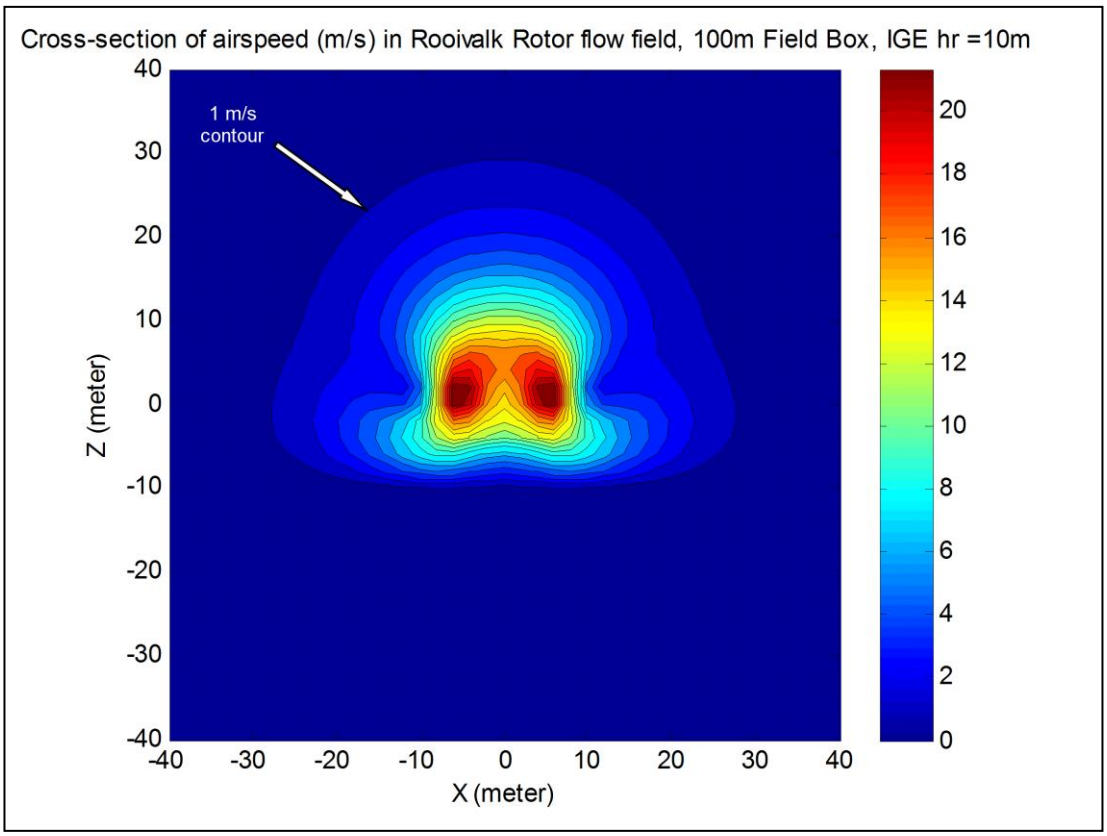


Figure 118. Rooivalk downwash: IGE hover, 100 m Flight Box, Height=10m [A Landman, 2014]

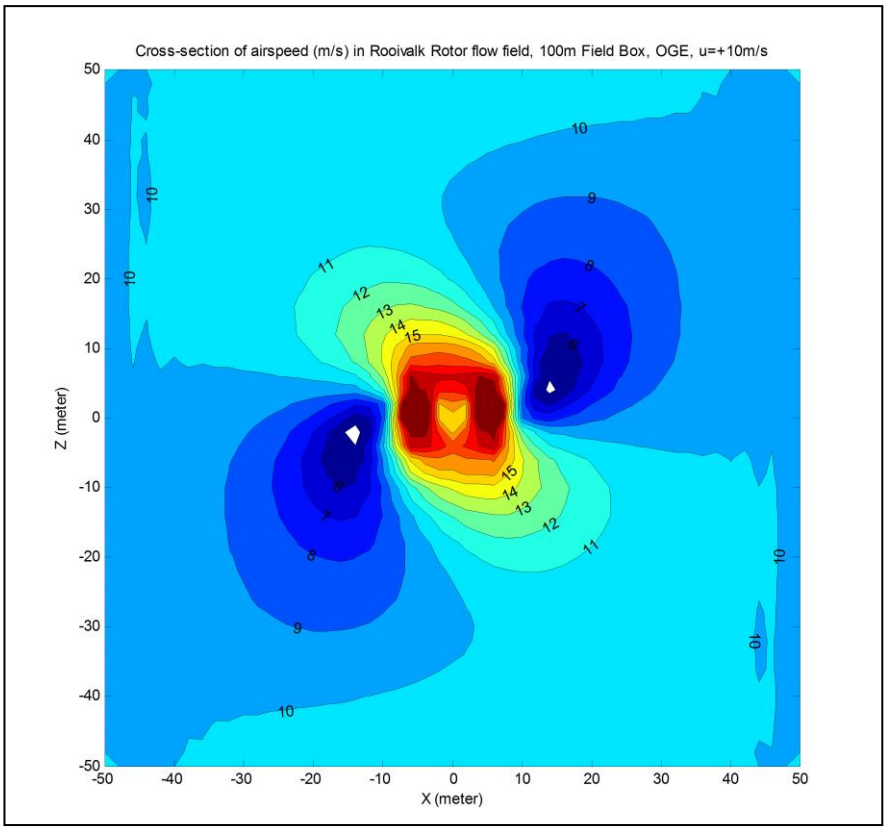


Figure 119. Rooivalk downwash: OGE hover, Wind=10 m/s, 100 m Flight Box [A Landman, 2014]

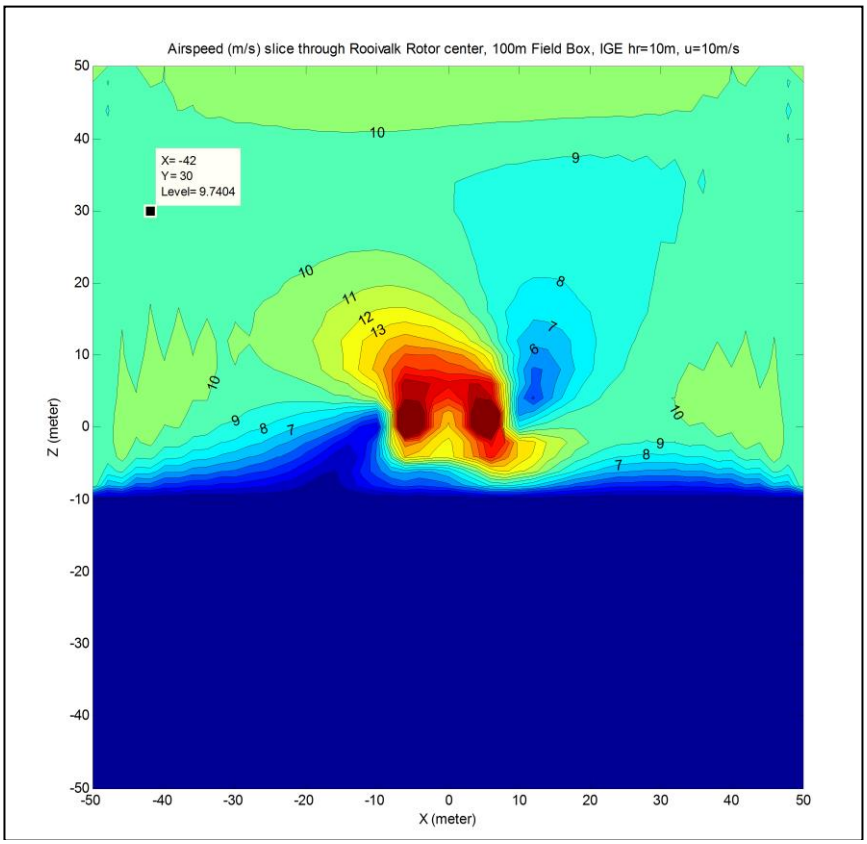


Figure 120. Rooivalk downwash: IGE hover at 10 m, Wind=10 m/s, 100 m Flight Box [A Landman, 2014]

4.6 Summary (Develop General Theories)

This chapter was used to execute the fourth Scientific Method Cycle of a family of nested Scientific Method Cycles that yield the concept of a Precision Rocket Launching System. To implement the PRLS concept suitable compensation must be introduced for the error mechanisms resulting in the poor ballistic performance of standard Free Flight Rockets (FFR). To achieve this, the errors caused by these mechanisms must be predicted in real time. The purpose of this chapter was to investigate the feasibility of developing a downwash algorithm that can calculate the downwash of a helicopter in real time. Real time operation was defined as an iteration rate of at least 10 kHz.

Photographs of the shape of the downwash flow fields of a number of helicopters as revealed by dust and water droplets trapped in the downwash flow fields of these helicopters were presented as main observation evidence. Additional observations in the form of images of the downwash flow field as aquired with a Light Detection And Ranging (LIDAR) sensor was also presented.

Probing questions were asked and used to formulate the Donut Hypothesis which postulates that the behaviour of the downwash of a helicopter is similar to the flow of current in a 3-dimensional transmission activated by a current source at centre. In such a system the current follows the path of least resistance, resulting in a flow field that resembles a donut in shape. Obstacles within the flow field resemble boundaries that will cause concomitant distortion of the flow field. The Donut Hypothesis also postulates that the Navier Stokes equations can be tailored to estimate the downwash of a helicopter in real time by embedding such an algorithm in contemporary electronics that employ massive parallelism.

Predictions based on the Donut Hypothesis were made. The first prediction stated that a version of the Navier Stokes equation can be derived that would be suitable for dividing the problem into many parts that can be executed in parallel. The second prediction stated that these parts can be executed in sequential fashion (yielding a slow process) and that such a simulation would confirm the donut shape of the downwash of a hovering helicopter. The third prediction stated that the simulation would predict a diameter of about 90 meter as observed for the brownout produced by a helicopter in the 10 ton class. The fourth prediction stated that the maximum wind speed in the downwash should be about 20 m/s.

To gather data for testing the Donut Hypothesis a tailored version of the Navier Stokes equations was developed that predicts the state of air in a single test volume, such that many of them can be linked in the form of a 3-dimensional array of test volumes to represent the computational domain or Flight Box of the problem. The dimensions of the test volumes can be chosen to yield any spatial sampling rate required. Similarly, the time increment of the process can be chosen to yield any temporal sampling rate required. A stable solver and suitable filters were added to remove aliases above Nyquist in both the temporal and spatial domains.

Having performed the research of Chapter 4 the following conclusions and theories are now formulated:

a) At least one tailored version of the Navier Stokes equation exists which can be used to estimate the state of air within a single test volume in a stable, repeatable and predictable fashion. This yields a basic building block or cell, multiples of which can be linked in the form of a 3-dimensional array to yield a computational domain of any size. Sampling rates in both temporal and spatial domains can be adjusted to suit the application.

b) Communication is only required between adjacent cells so that the array can be expanded without bound and without incurring a communication problem.

c) Cells operate as cellular automata and hence, can literally all run in parallel. This makes possible the use of massive parallelism offered by contemporary electronics to achieve the enormous throughput as called for by the PRLS hypothesis.

d) Provided a suitable electronic circuit can be constructed, it should be possible to manipulate individual cells within the array such as to introduce boundary conditions that represent the shapes and movements of objects within the computational domain. This mechanism should be suitable for simulating the dynamic behaviour of a complete system comprising Major Components such as air, main rotor, fuselage, engines, rockets and ground as illustrated in Figure 20. This assumption is likely to have a caveat in that the iteration rate of the array must be well in excess of the maximum frequencies describing the movements of the Major Components.

e) It should be possible to construct an electronic circuit such that the array can run standalone while boundary conditions are imposed asynchronously by an external model and a sub-set of the result contained within the array extracted asynchronously for external use (such as a ballistics algorithm).

5 The Precision Rocket Launcher (Cycle 5)

"Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world". Albert Einstein.

This chapter introduces the fifth Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. In order to explain the observed coning behaviour of FZ90 Rockets launched from Rooivalk AH-2A Chapter 2 formulated the Coning Hypothesis and concluded that the downwash of the helicopter must be calculated with high accuracy and in real time for the PRLS concept to work.

Chapter 4 investigated the feasibility of using the Navier Stokes equations for calculating the downwash of a helicopter in real time and in doing so derived the Taylored Navier Stokes equation and a stable method for solving it. This yielded an algorithm which can be duplicated into a large number of cells that can be arranged in a 3-dimensional array called the **Gas Flow Model** which can be configured to calculate the flow field of any Flight Box to any desired accuracy. Due to its structure, the **Gas Flow Model** was shown to be suitable for implementation on a massively parallel scale.

This chapter formulates the Automata Hypothesis which states that it should be possible to contrive a **Gas Flow** process (implementation of the **Gas Flow Model**) that runs independently while being steered by a **Commutation** process which considers the boundaries, excitations and constraints posed by multiple bodies such as the main rotor, fuselage, rocket launchers, FZ90 Rockets and ground that may be present in the Flight Box. The Automata Hypothesis also states that it should be possible to contrive a **Probing** process that extracts the solution from the **Gas Flow** process in concurrent fashion and a **Grid Hop** process that allows the system to analyze consecutive volumes of space.

Figure 121. M159 Rocket Launcher with REU fitted on Rooivalk AH-2A [A Landman, 2014]

Finally, the Automata Hypothesis states that it should be possible to construct a physical circuit using contemporary electronics packaged in a Precision Rocket Launcher with dimensions similar to the M159 Rocket Launcher as used on Rooivalk AH-2A and illustrated in Figure 121, such a to have a throughput of at least 0,601 Petaflop for implementing and running the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes in real time (10 kHz iteration rate).

Note: *The **Gas Flow** process forms part of the **Space** block of the **Rocket Launching System Model** as illustrated in Figure 20. The **Commutation** process forms a collection of interfaces between the **Space** block and other blocks such as the **Fuselage** and **Main Rotor** blocks. Similarly, the **Probing** process forms the interface between the **Space** block and a **Ballistic Algorithm** block.*

Note: *Combined, the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes form a partial build of Figure 20 referred to as the **Downwash Model** for the purposes of this study.*

The observational evidence presented to formulate the Automata Hypothesis includes a wide range of topics such as the outline and dimensions of the M159 Rocket Launcher, internal architectures of Central Processing Units (CPU), Graphic Processing Units (GPU) and Field Programmable Gate Arrays (FPGA) as well as the apparent limitations posed by the Amdahl and Gustafson laws.

Questions are asked as to what circuit and what type of device represent the best approach for implementing the **Downwash Model** in real time. Three fundamental constraints of such a circuit namely the Processor Bottleneck, Memory Bottleneck and Interface Bottleneck are identified and used to select the most suitable circuit – an array of FPGA devices arranged as a 2-dimensional array or sheet rolled around the exterior of the M159 Rocket Launcher. Such a circuit allows for the decentralization of processing, memory and interfaces. It is also well suited for parallel implementation on a massive scale.

Some predictions based on the Automata Hypothesis are made, the main prediction being that it is possible to package a 25x25 array of top-end FPGA's together with associated electronics into the Precision Rocket Launcher and that it is possible to commutate, probe and grid hop this array in real time. This prediction is tested by performing a detailed design of the Precision Rocket Launcher.

Note: *The hardware design of the Precision Rocket Launcher as presented in this chapter required many manhours to perform. Such a hardware design was required in order to prove the Automata Hypothesis. As such, it produced a large quantity of detail which is not repeated in this study. Only the main concepts used and conclusions reached during the design are described here.*

Note: *This chapter formulates a process description of the **Downwash Model** and then uses this process description to design a physical mechanism (or enabler as in Icam DEFinition for Function Modeling terms) for implementing the process. Programming of the physical mechanism (i.e. formulation of an actual circuit) to embed the **Downwash Model** occurs in Chapters 6 and 0.*

This chapter makes the following contributions to the knowledge database:

- It lays the foundation for relating the performance of the Array Processor to the Amdahl and Gustafson laws, the implications of which are developed further in Chapters 6, 7 and 8.
- It provides a process description of how such a device should operate.
- It provides a description of the physical design of a real Precision Rocket Launcher, how an Array Processor can be embedded in such a device and what limitations such a design places on the Array Processor.

5.1 Contemporary Electronics (Make Observations)

The **Gas Flow Model** comprises a 3-dimensional array of cells with each cell containing a copy of the algorithm as described in paragraphs 4.5.4 thru 4.5.8. Each cell talks to its nearest neighbours but otherwise operates in complete isolation. For the algorithm to work all the cells must work in unison and the nearest neighbor communications must occur at the same time, so the network effectively forms a 3-dimensional array of cellular automata that operates in lockstep. However, what circuit and component type would be most suitable to implement it with?

Contemporary electronics offers four device types that can be used to implement such a circuit namely Central Processor Units (CPU), Graphic Processor Units (GPU), Field Programmable Gate Arrays (FPGA) and Application Specific Integrated Circuits (ASIC). Note that for the purposes of the discussion here a Digital Signal Processor (DSP) is considered a special CPU optimized for performing operations such as the Fast Fourier Transform (FFT).

5.1.1 Central Processor Units

CPU's are a special version of a Turing engine [81]. Instead of a tape they employ Random Access Memory (RAM) using an architecture as originally described by *von Neumann* [78] in which an Arithmetic and Logic Unit (ALU) executes instructions fetched from Random Access Memory via a Data Bus and Address Bus. The ALU uses the same Data Bus and Address Bus to also read/write data from/to RAM.

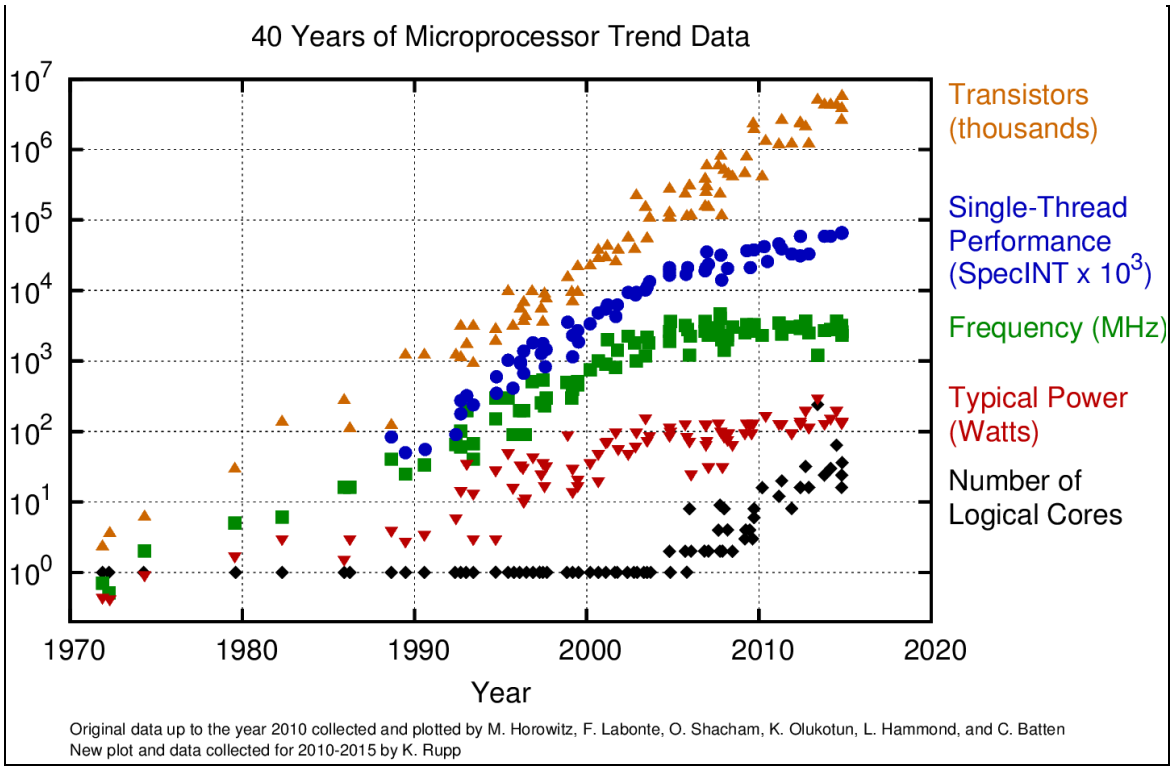


Figure 122. Evolution of CPU characteristics over past 40 years [Rupp, 2015]

As described by *Thornton* processors with a Harvard architecture is an improvement of the von Neumann architecture in that it uses a dedicated Data Bus and Address Bus to fetch instructions from RAM and another Data Bus and Address Bus to read/write data from/to RAM [79]. However, both machines share a common characteristic in that instructions are fetched and executed in sequential fashion, creating a constriction that *Backus* [80] called the **von Neumann Bottleneck**. This is a fundamental limitation of CPU's which can be mitigated (not avoided) to a

degree by things such as increasing the word width, increasing the clock speed, pre-fetching instructions in anticipation of what the ALU might do (pipelining) and using 3-dimensional or tri-gate transistors to reduce power consumption and increase switching speed.

Most CPU improvements associated with the scaling scheme of *Dennard et al* [83] have been exploited to the maximum as calculated by *Rupp* [82] and illustrated in Figure 122. For example, clock speed for air-cooled devices has tapered out at about 4 GHz, power at about 200 Watt and Single-Thread Performance is about to taper out at 10^5 SpecINT x 10^3. In contrast, the number of transistors available to implement the CPU is still increasing in accordance with *Moore's* law [84]. *Kurzweil* [85] argues that this trend is not bound to stop soon, since Integrated Circuits are but the fifth of a sequence of computing paradigms as illustrated in Figure 123 – others (such as quantum computers) are bound to follow.

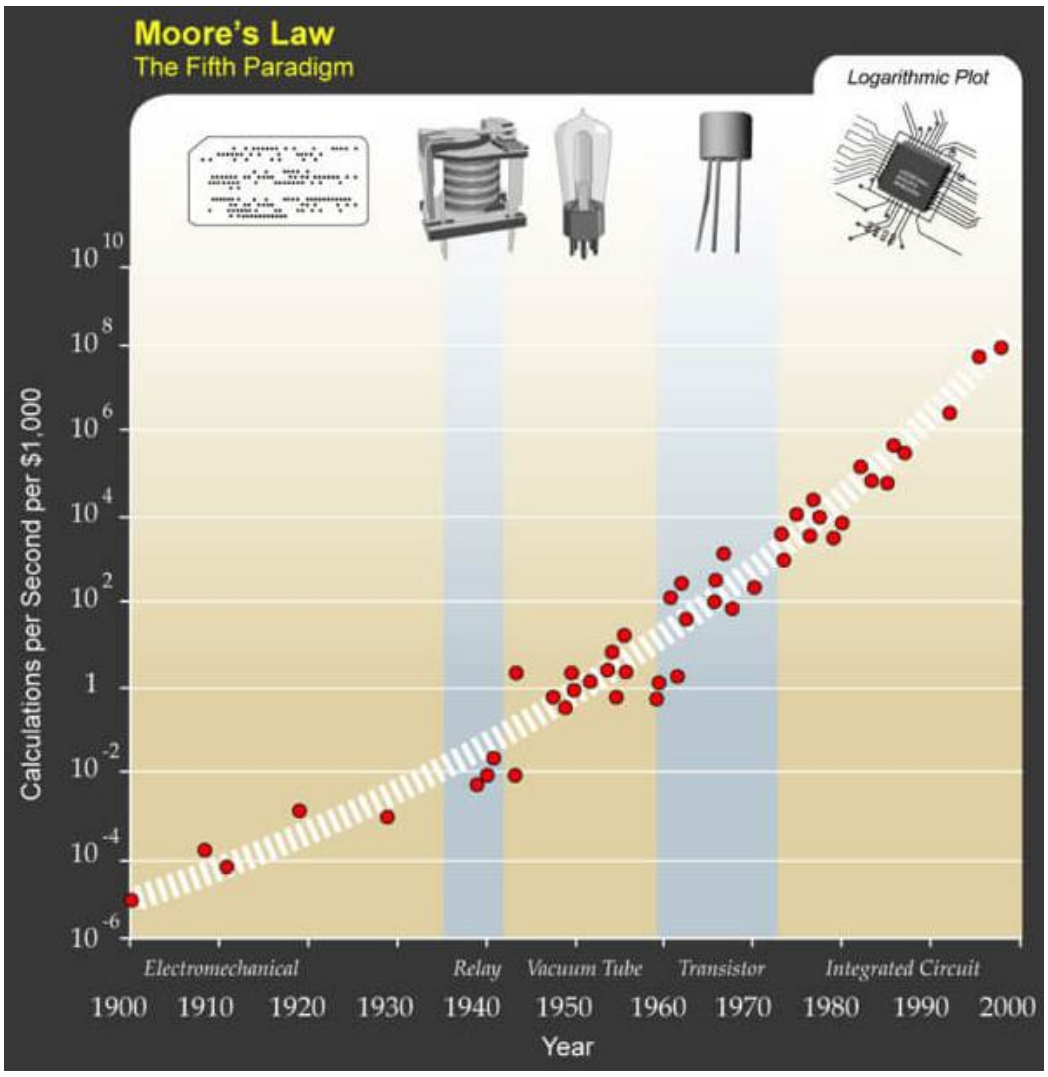


Figure 123. Integrated Circuits as the fifth computing paradigm [Kurzweil, 2005]

Note from Figure 122 that since 2005 the extra transistors were not used to build more powerful single-core processors as such but multi-core processors in a bid to mitigate the von Neumann Bottleneck which makes these devices subject to the limitations of the *Amdahl* [86] and *Gustafson* [87] laws. As at 2018 contemporary CPU's spanned a wide range of cores such as the *ProLiant ML110 Gen10* which contains 4 cores packaged in 1 device up to the *Integrity Superdome X* which comprises 384 cores packaged in 16 separate devices.

5.1.2 Graphic Processor Units

Graphic Processor Units represent the multi-core CPU concept pushed to the extreme. These devices can contain several thousand cores, each being a von Neumann processor with a limited instruction set. Note that such a solution is perfectly acceptable for implementing the Array processor as envisaged by this study because the algorithm as developed in paragraphs 4.5.4 thru 4.5.8 only requires basic operations such as add, subtract, multiply and divide.

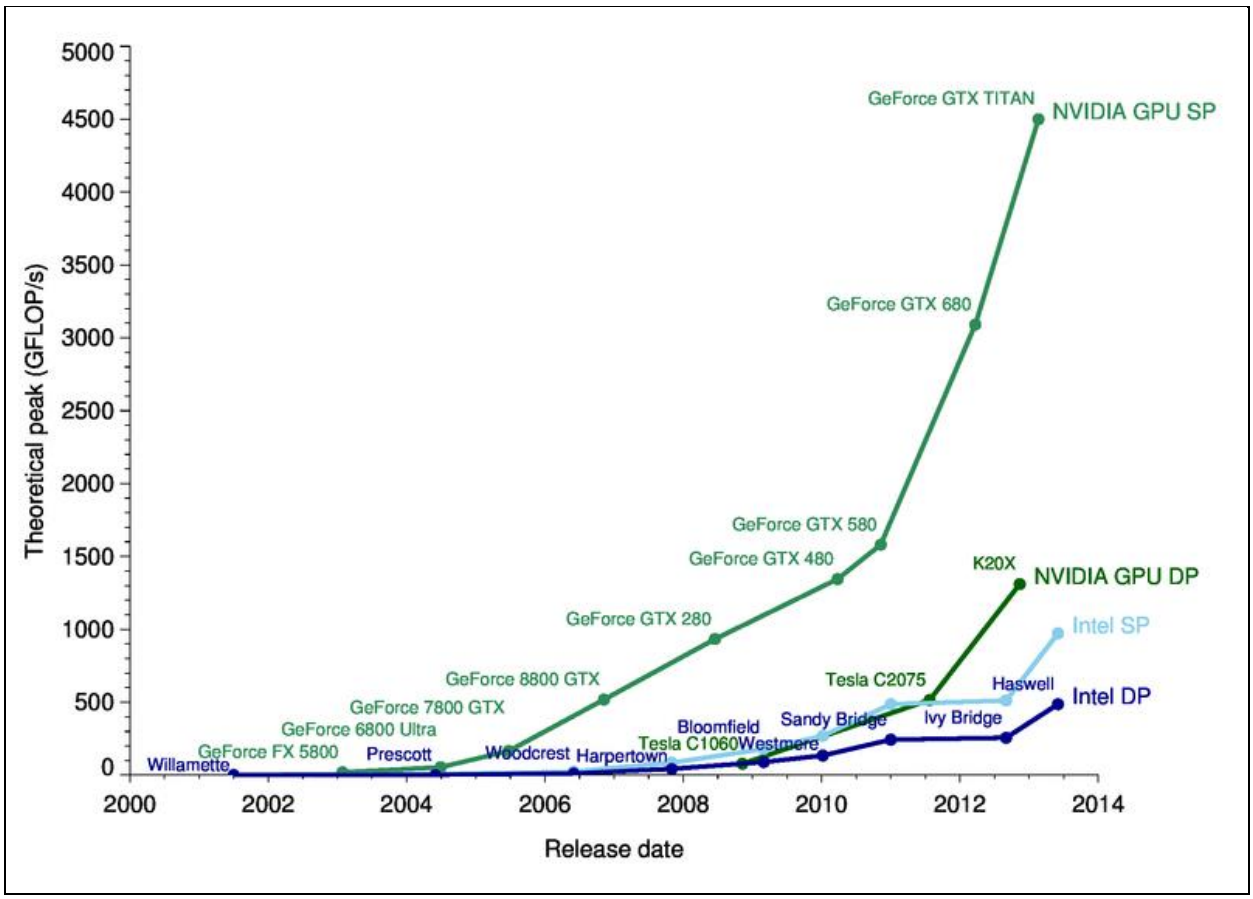


Figure 124. GPU versus CPU performance comparison [Galloy, 2013]

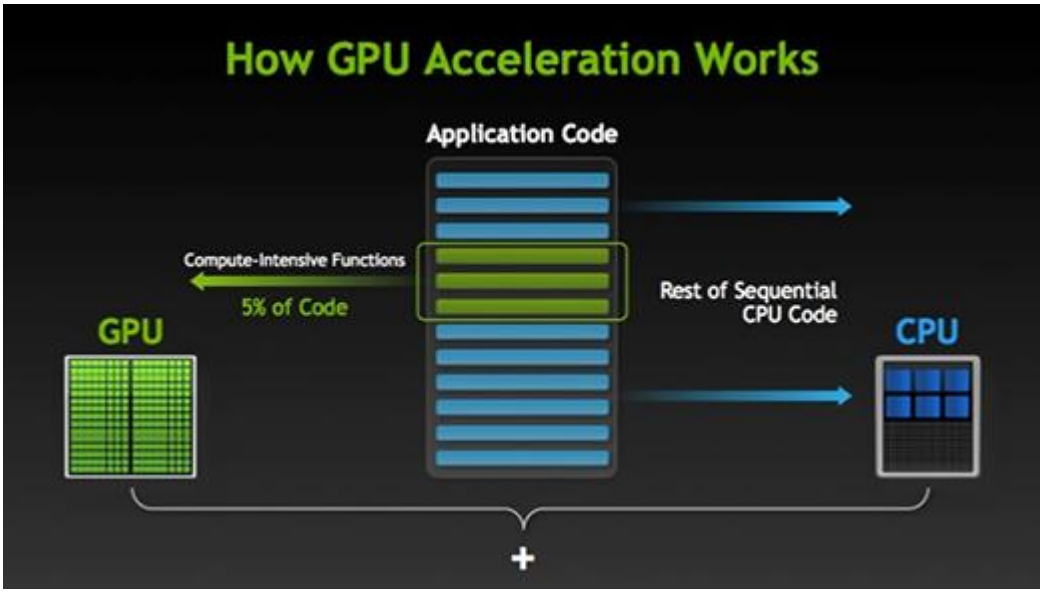


Figure 125. Allocating parallel and sequential functions to GPU and CPU [NVIDIA, 2012]

The evolution in peak performance (single precision and double precision FLOPS) of GPU's versus CPU's as determined by *Galloy* [88] is illustrated in Figure 124. The advantage of the massive parallelism of GPU's is evident.

GPU's are difficult to program and are not general purpose machines like CPU's. They are dedicated processors that should operate in conjunction with a CPU in a so-called heterogeneous computing scheme as illustrated in Figure 125. In such an arrangement the GPU performs the computationally intensive part of the task while the CPU performs the housekeeping and/or sequential part of the task. This would be a suitable scheme for implementing the Array Processor as envisaged in this study except that an array of GPU's would be used. Note that such a scheme is similar to the multi-processor scheme as assumed by *Amdahl* [86] and and *Gustafson* [87].

As described by *Fatahalian* and *Houston* [89] as well as *Cook* [90] the power of a GPU stems from the fact that it is a so-called Single Instruction Multiple Data (SIMD) machine in accordance with the classification scheme as defined by *Flynn* [91]. The software running on a GPU is organized in terms of **threads** akin to tasks in Real Time Operating System (RTOS) terms. Threads are grouped into **warps** within which all the threads run concurrently on separate von Neumann processors (cores) which use the same program counter and execute the same code on their own data sets, thus gaining the advantage of massive parallelism and avoiding duplication of instruction decoding logic for each von Neumann processor. Depending on the program path followed by each thread, warps can be created, suspended and scheduled by hardware, ensuring maximum loading of all the von Neumann processors all the time.

As described by *Cook* [90] the memory model and programming model required to realize such a scheme is complex. Essentially, each thread has access to its own local memory, threads within a warp have acces to shared memory (which enable threads within a warp to exchange information) and all threads have access to global memory (which enable all threads to exchange information). Global memory can also be accessed by the Host CPU which can schedule a thread to copy code/data from its own host memory to global memory of the GPU before triggering the GPU to run the code.

Data accesses as described above cannot all happen at the same time since the GPU has a limited number of internal busses, albeit some of which are 256 bits wide to achieve data throughputs of hundreds of GByte/sec. For this reason the GPU is most efficient when threads of a warp access contiguous memory locations. It also causes memory accesses between local memory and a thread to be faster than memory accesses between shared memory and threads of a warp. For the same reason memory accesses between shared memory and threads of a warp is faster than memory accesses between threads and global memory. Finally, memory accesses between global memory and host memory is the slowest of them all because it occurs via the PCI-E bus at a rate of 5 GB/sec.

5.1.3 Field Programmable Gate Arrays

The basic concept of the FPGA is described very well by a set of diagrams produced by *Vaughn Betz* of the University of Toronto and illustrated in Figure 126, Figure 127 and Figure 128. As shown, it is comprised of a matrix of Configurable Logic Blocks (CLB) interspersed within an array of routing channels (wires) that can be configured with programmable switches located at crossovers. Each Configurable Logic Block is comprised of a Look Up table (LUT) and a D-type flip-flop, producing one registered or one un-registered output as illustrated in Figure 128.

Inspection of Figure 126 and Figure 128 shows that provided the FPGA is equipped with enough resources, such a circuit can be configured to calculate any computable function. This can be achieved by configuring the circuit to yield a wide range of building blocks such as logic gates, registers, memory, adders, multipliers and connecting these blocks to form complex circuits including complete CPU's. Herein lies the power of the FPGA – it can be configured to create a

dedicated circuit that computes only what is required and does it with the entire circuit operating in parallel.

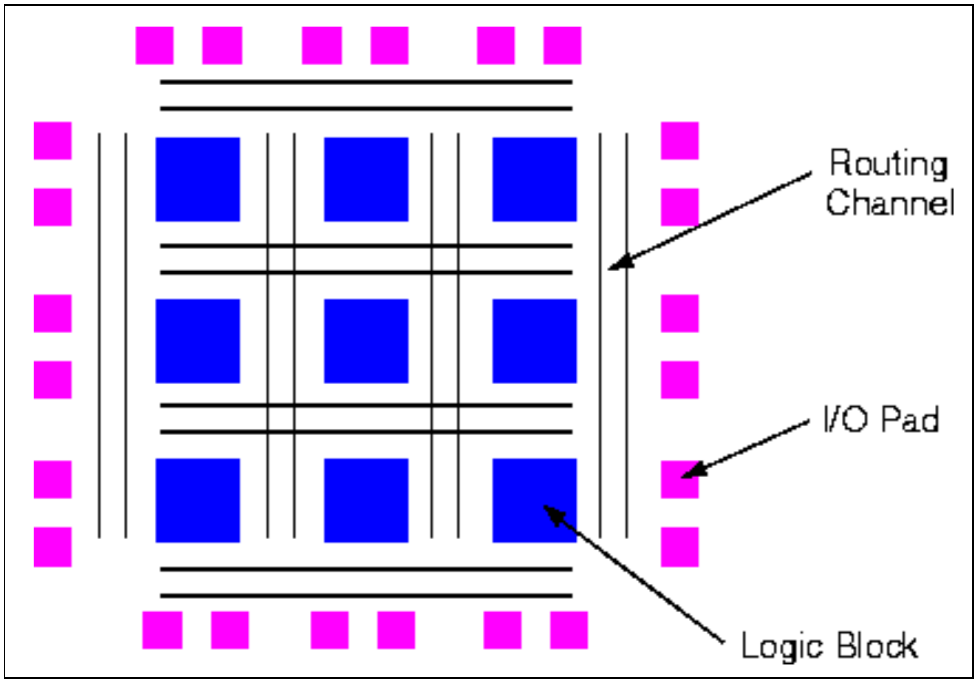


Figure 126. Basic FPGA architecture [Betz, 2018]

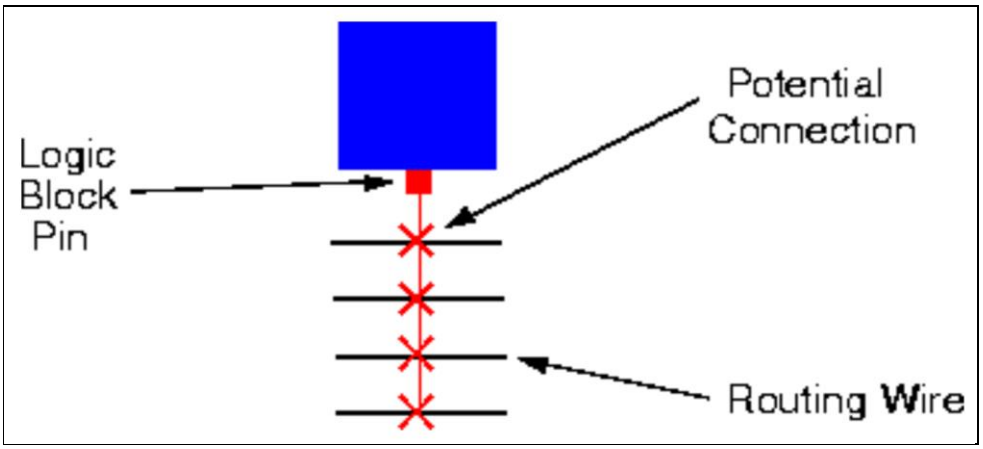


Figure 127. Connections into Configurable Logic Blocks [Betz, 2018]

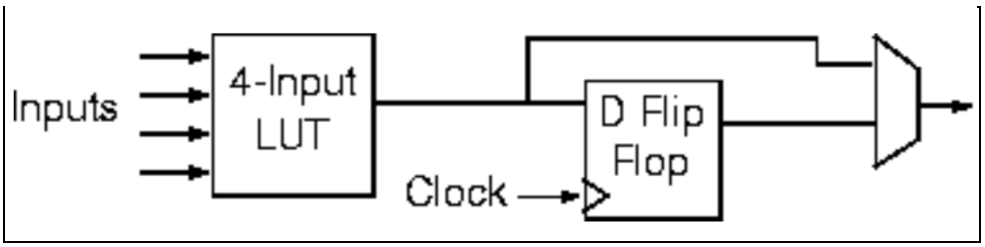


Figure 128. Basic structure of Configurable Logic Blocks [Betz, 2018]

Moore's law has also been at work in increasing the number of transistors used to implement contemporary FPGA's. For example, the STRATIX 10 device family made by Intel contains 30 billion transistors [92]. This translates into a device with 5,5 million Configurable Logic Blocks and 1152 hardware multipliers that yield a throughput of 10 TeraFLOPS (single precision) [93].

Note that contemporary FPGA's are available in a wide and diverse range of devices that may include dedicated (hard-wired) resources such as timers, A/D converters, D/A converters, I/O buffers, multipliers and ARM processors. It is also possible to embed any number of soft processors in the FPGA such as the NIOS II processor in case of Intel devices [95]. For a given FPGA application, the problem then becomes choosing a device that yields the best match for the problem at hand.

Note: *The problem of choosing the best device is complicated by market forces that cause FPGA manufacturers to favour certain configurations that satisfy the requirements of big markets such as the automotive and cellphone industries. For many applications this means resources that cannot be used, translating into wasted resources with concomitant space, mass, power and financial penalties. There is no guarantee that a device exists that would yield a good match for constructing the Array Processor as envisaged in this study.*

5.1.4 Implications of the Amdahl and Gustafson laws

Predicting the performance of multi-processor computers was the subject presented by *Gene Amdahl* during the American Federation of Information Processing Societies (AFIPS) Spring Joint Computer Conference of 1967 [86]. In his paper *Amdahl* made a case for the superiority of single-processor architectures versus multi-processor architectures due to a number of "complications". In particular, he quotes the following:

> *"The effect of each of these complications is very severe on any computer organization based on geometrically related processors in a paralleled processing system".*

Although *Amdahl* wrote his paper in rather vague terms it would seem that he was talking about something very similar to the Array Processor as considered in this study. Judged by Amdahl's law, the odds of achieving the performance as envisaged in this study for the Array Processor does therefore not appear good at first glance. *Madden* [94] warns in his paper "*Parallel Computing: The Elephant in the Room*" that although parallel processing may seem to hold promise of unbounded speedup, it is mostly a false hope and a trap that many computer scientists have fallen into. This warning must therefore be heeded with the due diligence that it deserves.

It is evident from *Amdahl's* paper that the computer developers of the 1960's held the size of their application (together with associated housekeeping) constant irrespective of the available processing power. These assumptions can be expressed with an equation:

$$\begin{aligned} Speedup &= \frac{T_o}{T_n} \\ &\equiv \frac{\alpha\, T_o + (1-\alpha) T_o}{\alpha\, T_o + \left(\frac{1-\alpha}{N}\right) T_o} \\ &\equiv \frac{1}{\alpha + \left(\frac{1-\alpha}{N}\right)} \end{aligned}$$

Equation 112

With T_o the time taken for a single-processor to complete a given task, T_n the time taken to complete the same task with n similar processors and α the fraction of T_o required to perform the

housekeeping function of the task. Inspection of Equation 112 shows that the speedup of such a system approaches *1/α* for *N* large. According to *Amdahl* the value of *α* amounted to 0,4 for "production runs" over a period of 10 years at the time that he presented his paper. He then concludes:

> *"In an entirely dedicated special purpose environment this might be reduced by a factor of two, but it is highly improbable that it could be reduced by a factor of three. The nature of this overhead appears to be sequential so that it is unlikely to be amenable to parallel processing techniques. Overhead alone would then place an upper limit on throughput of five to seven times the sequential processing rate, even if the housekeeping were done in a separate processor."*

In other words, *Amdahl* claimed that the housekeeping function limits the performance improvement of a multi-processor computer to about 5 times that of a single-processor computer assuming *α* to have a value of 0,4. Any attempt to improve the performance of a computer by employing massive parallelism would soon be stifled unless the computer responsible for housekeeping would be improved with a similar amount.

Some 21 years after *Amdahl* wrote his paper *John Gustafson* [87] wrote another paper titled "*Reevaluating Amdahl's Law*" in which he states:

> *"We now have timing results for a 1024-processor system that demonstrate that the assumptions underlying Amdahl's 1967 argument are inappropriate for the current approach to massive ensemble parallelism".*

Quoting some real-world applications *Gustafson* showed a clear divergence between reality and what Amdahl's law predicted for those applications. It is evident from *Gustafson's* paper that designers of the late 1980's scaled their application in accordance with the available processing power while leaving housekeeping constant. *Gustafson* compared the real-world scenario with the *Amdahl* scenario by asking how much time a serial machine would require to perform a task currently performed by a parallel machine. These considerations can be expressed with an equation:

$$\begin{aligned} Speedup &= \frac{T_o}{T_n} \\ &\equiv \frac{T_s + N\,T_p}{T_s + T_p} \\ &\equiv \frac{T_s + N\left(T_n - T_s\right)}{T_n} \end{aligned}$$

Equation 113

With *N* the number of cores employed in a multi-processor, T_n the time taken by the multi-processor to complete a given task, T_o the time that a single core would take to complete the same task, T_s the time that a single core would take to complete the serial part of the task and T_p the time that the multi-processor takes to complete the parallel part of the task.

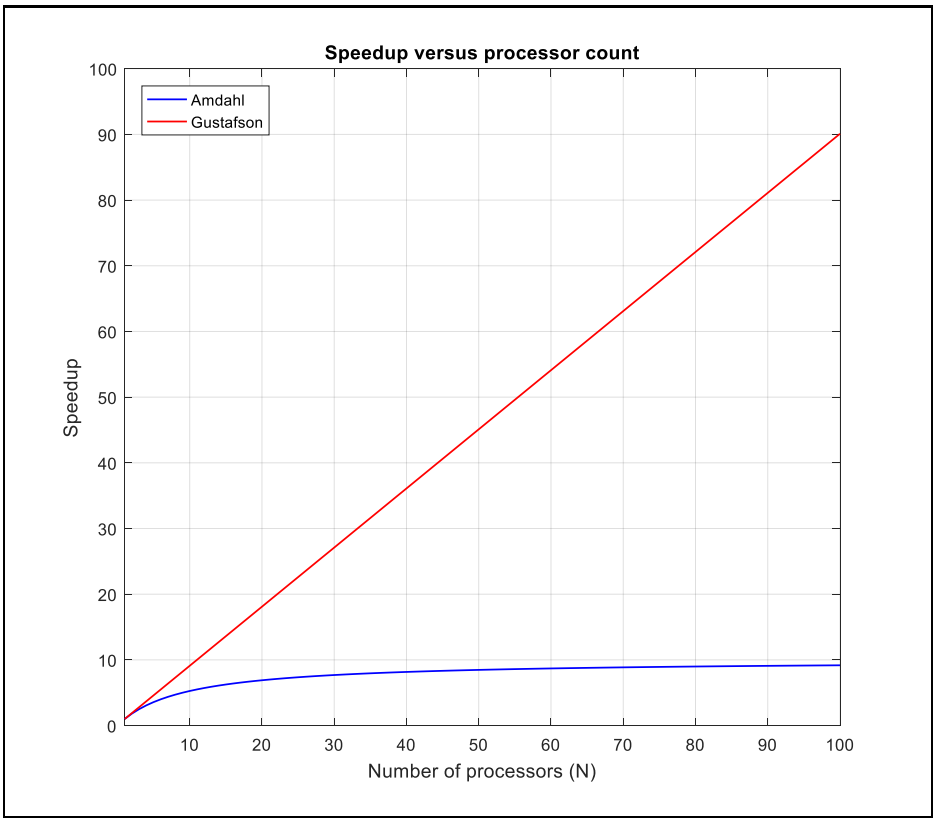


Figure 129. Amdahl and Gustafson laws assuming $\alpha = 0,1$; $T_s = 1$; $T_n = 10$ [A Landman, 2018]

It is clear that whereas the *Amdahl* curve levels out at *1/α* the *Gustafson* curve progresses linearly with slope $\left(\frac{T_n - T_s}{T_n}\right)$ which approaches the value 1 for $T_s << T_n$. In other words, the slope of the *Gustafson* curve levels out at one for large numbers of cores. The main reason for this major difference is that *Amdahl* assumed the household fraction to be directly proportional to the overall task while *Gustafson* showed it to be constant. It is evident that provided the housekeeping function required to run the **Downwash Model** can be kept constant the limitation of *Amdahl* can be avoided.

5.2 Which is Best? (Think of interesting questions)

The comparison of contemporary electronic devices suitable for implementing the Array Processor as presented in the previous paragraph revealed two extremes. Located at the one extreme are CPU's that re-use the same circuit at the expense of overall speed of computation. Located at the other extreme are FPGA's that implement everything in parallel at the expense of real estate usage. Which approach is best? This question has been the subject of numerous articles available in the open literature. *Parker* [97] approached the question by calculating the absolute maximum throughputs of a typical range of DSP, GPU and FPGA devices. He found 160 gigaflops for the TMS320C667 DSP running at a clock frequency of 1.25 GHz, 3520 gigaflops for the NVIDIA Tesla K20 GPU running at a clock frequency of 706 MHz and 1520 gigaflops for the 10AX066 FPGA running at a clock frequency of 450 MHz. *Jones, Powell, Bouganis & Cheung* [96] approached the question by comparing the performance of the Convey HC-1 FPGA server card and the Nvidia GTX285 GPU performing some common High-Productivity Computing System (HPCS) applications and found the GPU to outperform the FPGA card in four of the five cases. They also tried to answer the question in terms of the "time

to solution" criteria as adopted by the Defense Advanced Research Projects Agency (DARPA), considering both their own work and the work done by *Tsoi & Luk* [98] who concluded that an n-body simulation implemented on FPGA using customised hardware and firmware can run twice as fast as the same application running on a GPU. Countering this fact *Jones, Powell, Bouganis & Cheung* [96] note that this option is only possible for hardware developers that must be "both sufficiently equipped and capable to develop the necessary firmware for FPGA-based HPCS". It is clear that assessment of this question depends on a variety of considerations that may or may not be applicable to a given situation. In the case of the PRLS the main considerations are as discussed below.

5.2.1 Implementing the Array Processor with CPU

Parker [97] used device TMS320C667x of Texas Instruments as a typical multi-core DSP. This device has a peak performance of 160 gigaflops running at a clock frequency of 1,25 GHz and is typical of the type of device that would be required should the Array Processor as envisaged by this study be implemented with an array of CPU's. Chapter 7 calculates the throughput of the Array Processor as 0,601 petaflops single precision. Hence, assuming that a circuit comprising an array of TMS320C667x devices can be designed so that each device runs at 90% of peak performance the Array Processor would require 4174 of these devices to achieve a throughput of 0,601 petaflops. To pack such a large number of devices within the physical constraints of a Line Replaceable Unit (LRU) like the M159 Rocket Launcher is not a viable solution, irrespective of how much power such a circuit might consume. *Martinez, Bond & Vai* [99] report a factor 200 improvement of FPGA over conventional software. In addition, the circuit would have very poor reliability due to the large number of devices used.

5.2.2 Implementing the Array Processor with GPU

Figure 130 and Figure 131 show the TESLA V100 card with P8C406000 device – the newest and most powerful GPU manufactured by NVIDIA as at 2018. This GPU contains 21 billion transistors, yielding 5120 CUDA cores with a throughput of 15,7 teraflops single precision. The GPU is mounted on a PCB on top of a steel plate as illustrated in Figure 131 and is fitted with two connectors – one for connecting to a host CPU via the PCI-E bus and another for connecting to other TESLA V100 cards via the NVLink bus.

Figure 130. NVIDIA TESLA V100 card with P8C406000 GPU, top view [Paul Alcorn, 2017]

Figure 131. NVIDIA TESLA V100 card with P8C406000 GPU, bottom view [Paul Alcorn, 2017]

Assuming that a circuit comprising an array of TESLA V100 cards can be designed so that each core runs at 90% of peak performance the Array Processor would require about 43 of these cards to achieve a throughput of 0,601 petaflops. Since a single TESLA V100 card dissipates 300 Watts, such an Array Processor would dissipate a total of 13 kW – a big heat load to dissipate within the bounds of an LRU similar to the M159 Rocket Launcher.

The TESLA V100 card has a form factor which NVIDIA refers to as "SXM2". No information on the exact sizes of the SXM2 form factor could be found in the open literature, but the surface area of the silicon die is quoted as 815 mm^2 in the open literature. This means that the silicon die must be roughly 30 x 27 mm in size. Scaling the image of Figure 130 the dimensions of the TESLA V100 card must therefore be about 108 x 52 mm.

The M159 Rocket Launcher has a length of 1687 mm and a diameter of 402 mm so that its outer surface area (longitudinal) is about 2,13 m^2. Assuming the Array Processor to be located on the longitudinal outer surface of the M159 Rocket Launcher (where it would be the easiest to remove the large quantity of heat generated) and implementing the Array Processor with TESLA V100 cards would cover an area of about 0,242 m^2 which is 8,8% of the available area. It is evident that it may well be possible to use 43 off TESLA V100 cards to implement the Array Processor as envisaged by this study. However, such a solution would have a number of problems:

- Drawing 13 kW of electrical power from the power generation system of Rooivalk AH-2A is not desirable from an aircraft perspective since it may well exceed the power and current limits of the alternators and wiring harness. This problem would be even worse if the aircraft is loaded with two Rocket Launchers (the normal configuration) to ensure the Centre of Gravity of the aircraft remains within limits. Depending on mission considerations the aircraft can also be loaded with four Rocket Launchers in which case the power consumed would be way too much.
- Removing 13 kW of heat from a rocket launcher similar in size to the M159 Launcher would demand a complex solution with concomitant volume and mass penalties. Such a solution may also result in hot parts that could generate IR signals enabling MANPADS missiles to lock onto the aircraft.
- As shown in Figure 153 and Figure 154 each core located at the border of a given device has to exchange the values of ten internal parameters to the adjacent core located at the border of the adjacent device. In the case of an Array Processor based on TESLA V100 cards these exchanges would have to be done via the PCI-E bus (which runs at 5

gigabyte/sec) because the NVLink bus can only be used to interconnect 6 off TESLA V100 cards in a limited number of configurations (running at 300 gigabytes/second). Assuming the Array Processor to comprise 525 x 525 cores (refer paragraphs 5.5 and 7.6) each TESLA V100 card would house an array of 80 x 80 cores. This means a total of $80*4*525*10*2=3{,}36.10^6$ data words (each comprising 32 bits) would have to be transported bidirectionally over the border of each TESLA V100 card during each major cycle. At an iteration rate of 10 kHz this would require a data rate of 134,4 gigabyte/second which is well in excess of the 5 gigabyte/second data rate of the PCI-E bus.

5.2.3 Implementing the Array Processor with FPGA

The above reasoning would seem to suggest the use of FPGA's which should dissipate less heat because they can be tailored to the exact needs of the Array Processor. Apart from the fact that they truly run in parallel, FPGA's can simultaneously solve the Processor Bottleneck, Memory Bottleneck and Interface Bottleneck through massive decentralization. To confirm that such a solution would be possible, consider the STRATIX 10 device as illustrated in Figure 132. This device family represents the most potent FPGA's manufactured by Intel as at 2018. In particular, device GX2800 from the STRATIX 10 family contains 11520 hardware multipliers (18 x 19 bit) which can yield a combined throughput of 9,2 teraflops [93].

Assuming that a circuit comprising an array of GX2800 devices can be designed so that the circuit runs at 90% of peak performance the Array Processor would require 73 of these devices to achieve a throughput of 0,601 petaflops. Device GX2800 is available in a number of packages, the smallest of which is a Ball Grid Array (BGA) package with dimensions 42,5 x 52,5 x 1,0 mm. The total surface area of such an Array Processor would therefore be about 0,163 m^2 which will occupy 13,1% of the outer surface of the M159 Rocket Launcher.

Figure 132. Typical specimen of Stratix 10 device family [Intel, 2016]

A model for calculating the power budgets for Stratix 10 devices in a range of typical applications is described in reference [100]. Applying this model and assuming the GX2800 device to be clocked at 800 megahertz (maximum clock frequency of STRATIX 10 family) with all multipliers active yields 124 Watt/device as estimate. Operating 73 of these devices under these conditions

would thus imply a total power requirement of 9,1 kW for the Array Processor. Construction of such a system would be more plausable from a thermal perspective .

Note: *Although this study showed FPGA's to be a viable building block for implementing the Array Processor, it also showed that even FPGA's are subject to the problem of unused resources. We will return to this topic in Chapters* 6 *and 7.*

Finally, *Amdahl* [86] mentions other constraints of multi-processors such as the memory bottleneck and I/O bottleck in his paper while *Gustafson* [87] makes no mention of these limitations. Should the Array Processor as envisaged in this thesis be implemented with FPGA's each core would have its own local memory so there would be no common memory that can cause a memory bottleneck as described by *Amdahl*. Similarly, the communication bottleneck will be avoided because nearest neighbor communications between adjacent cores located in adjacent FPGA devices can occur via dedicated SERDES links (device GX2800 contains numerous buffers to implement this type of link).

5.3 The Automata Hypothesis (Formulate hypothesis)

Having considered all the relevant evidence, we now postulate that it should be possible to invent a process description of the **Gas Flow Model** comprising a three-dimensional array of cells with each cell containing a copy of the algorithm as described in paragraphs 4.5.4 thru 4.5.8 in a circuit comprised of a number of interconnected FPGA devices. It should be possible to arrange this circuit in such a way that it acts like a network of cellular automata with each cell performing exactly the same function in lockstep with each other as clocked by a free-running state machine. By employing dual-port memory it should be possible to construct a **Commutation** process that concurrently manipulates the internal states of individual cells from external, such that the excitation and deflection effects of objects such as a main rotor blade, fuselage, ground and gas flows from engine exhausts can be factored into the flow field solution calculated by the array. It should also be possible to construct a **Probing** process that concurrently probes the solution as calculated by the array and a **Grid Hop** process that allows the system to analyze consecutive volumes of space. Finally, it should be possible to construct a physical circuit using contemporary electronics packaged in a Precision Rocket Launcher with dimensions similar to the M159 Rocket Launcher as used on Rooivalk AH-2A, such a to have a throughput of at least 0,601 petaflop for implementing and running the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes in real time (10 kHz iteration rate).

Note: *The throughput of 0,601 petaflop as stated above is calculated in paragraph 7.6.1. It is an example of many similar cross-feeds between the secondary Scientific Method Cycles as pursued under this study.*

5.4 Automata Predictions (Develop testable predictions)

The first prediction is that it would be possible to design a practical circuit (one that can actually be built with real components) of the **Gas Flow Model** running with a Major Cycle rate of at least 10 kHz.

The second prediction is that it would be possible to design a practical **Commutation** circuit (one that can actually be built with real components as integral part of the overall PRLS circuit) that can be used to manipulate the internal states of the **Gas Flow** circuit in real time without delaying the operation of the latter.

The third prediction is that it would be possible to design a practical **Probing** circuit (one that can actually be built with real components as integral part of the overall PRLS circuit) that can be used to probe the flow field solution in real time without affecting the operation of the **Gas Flow** and **Commutation** circuits.

The fourth prediction is that it would be possible to design the circuit such that the internal states of all cells can be shifted one cell position towards front, right, rear, left, top and bottom within one major cycle (0,1 msec).

The fifth prediction is that it would be possible to package the entire circuit as described above in a Precision Rocket Launcher with functions and geometry similar to the M159 Rocket Launcher, such that the design provides adequate cooling and environmental protection yet is simple and easy to construct.

5.5 Birth of the Beast (Gather data to test prediction)

How can a physical Precision Rocket Launcher (PRL) be constructed so that it not only carries and launches FZ90 Rockets like the M159 Rocket Launcher but also contains a large number of interconnected FPGA devices that implement a three-dimensional array of cells with each cell containing a copy of the Taylored Navier Stokes equation? To answer this question, consideration was given to a wide range of factors, the main ones of which are as described below, resulting in the PRL Physical Layout as illustrated in Figure 133 and the PRL Architecture Block Diagram as illustrated in Figure 134.

The basic physical layout of the PRL is that of a hexagonal “space tube” which contains 19 launch tubes with a large flexible printed circuit board rolled around its exterior. The circuit board interconnects a 5 x 5 array of **Array Processor Modules** (APM) that span 5 sides of the space tube and can be removed and replaced for programming and maintenance purposes. Each Array Processor Module contains a 5x5 array of top end FPGA’s. Overall, this arrangement forms a 25x25 array of top-end FPGA devices. Since device GX2800 from the STRATIX 10 family has a maximum throughput of 9,2 teraflops [93] it is evident that the overall array would have a maximum throughput of 5,75 petaflop should device GX2800 be used. This is an order of magnitude more than the 0,601 petaflop required (we will return to this topic in paragraph 7.2).

The top outer surface of the space tube contains the Host Processor, Power Supplies and NATO hooks for attaching the PRL to the Aircraft. The effective outer surface area of the PRL that houses electronics comprises 6 blocks, each 1,5 meter long and 0,24 meter wide. Of these, five blocks contain the Planar Array Processor (i.e. a total surface area of 1,8 m^2) and one the Main Electronics Unit.

The basic electrical layout of the PRL is that of a Host Processor controlling a number of intelligent front-ends through a high-speed Modelling Bus as illustrated in Figure 134. Some of the front-ends are dedicated Digital Signal Processors (DSP) that run dedicated software such as a main rotor model or a ballistic algorithm. Another front-end is a MIL-STD-1553B [101] interface which, together with a number of discrete and power lines, form a MIL-STD-1760A [102] interface with the Aircraft. Each launch tube is fitted with a micro-processor and associated electronics to form 19 Electro-Magnetic Launch Tubes. The Array Processor forms an intelligent front-end controlled by an Array Coordinator which drives a 5x5 array of Array Processor Modules and boundary blocks through a many-bit parallel Commutation Bus.

Note: *The PRL contains features such as Levitation Coils and Magnetic ID Sensors which are not described in this study. These features are the result of other studies as identified in Chapter 8.*

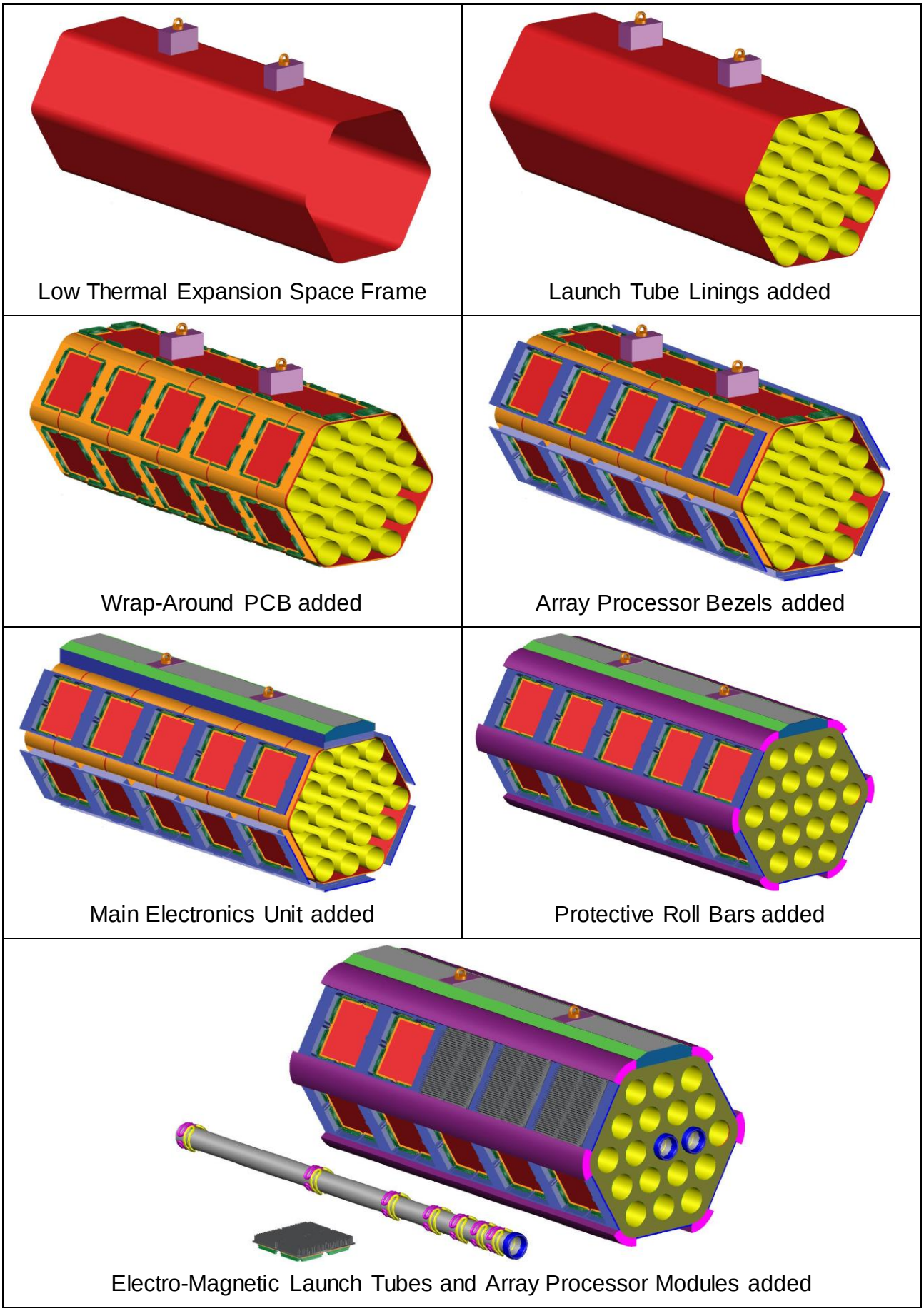


Figure 133. Physical Layout of the PRL [A Landman 2017]

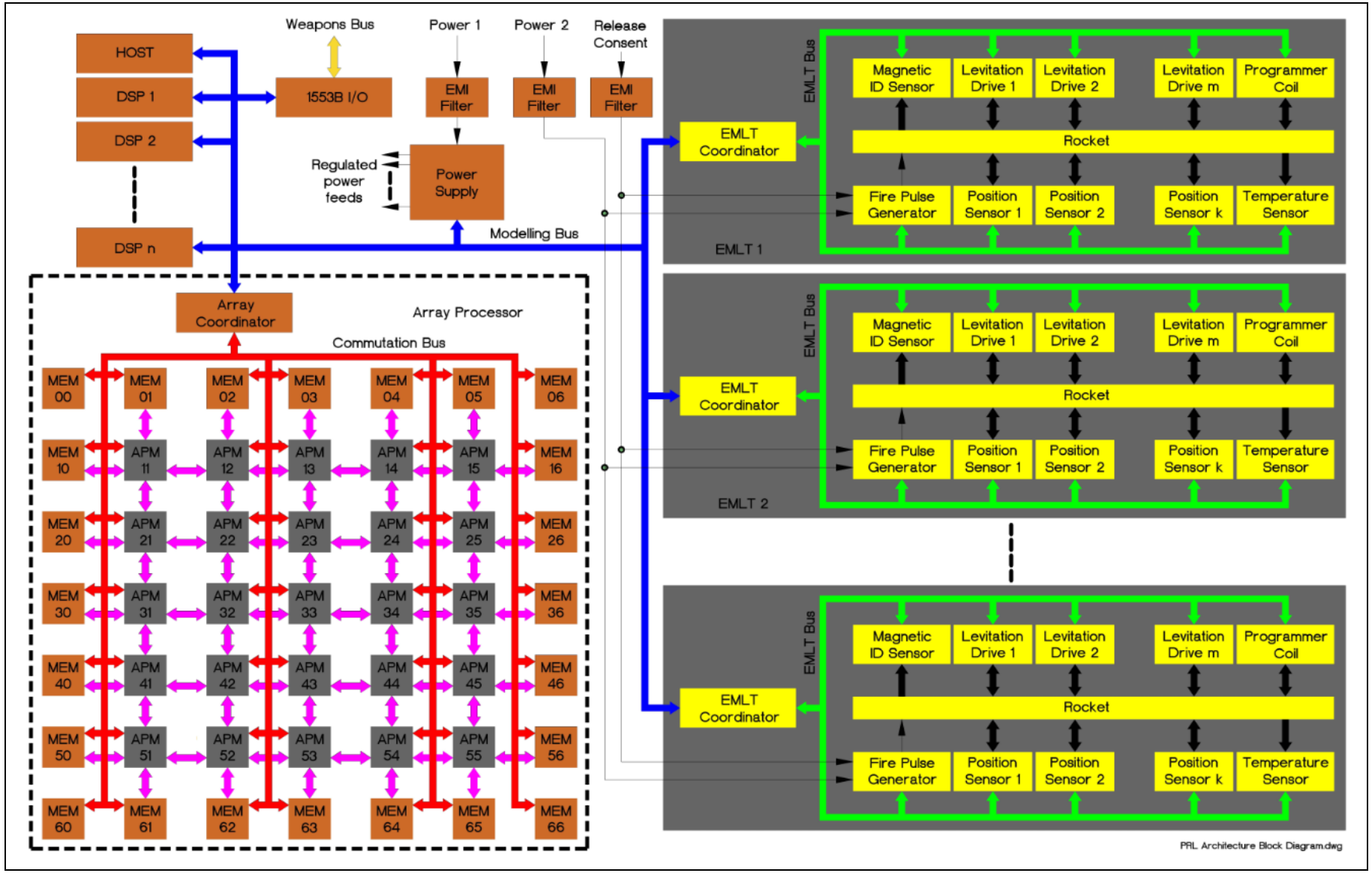


Figure 134. Architecture Block Diagram of the PRL [A Landman: 2017]

5.5.1 Re-Use versus Dedicated Use

The problem of finding a common measure for evaluating the performance of different technologies was described in paragraph 5.2. One factor common to all the technologies discussed is that each of them is implemented with equally large numbers of Field Effect Transistors (FET). Essentially, the differences between CPU, GPU and FPGA are to be found in how these transistors are applied. *Kurzweil* [85] showed that Moores Law is but part of a larger evolution in computational technology as illustrated in Figure 123. We assert that this evolution actually stretches over thousands of years and that it is accompanied by a repeating pattern of re-use that tends to be inversely proportional to the maturity of technologies (e.g. abacus to quantum computer) used for the computation.

The concept of re-use is based on a simple premise – the more scarce a given computation resource, the more such a resource will be re-used for the different computations required by a given application. For example, the Electronic Numerical Integrator And Computer (ENIAC) processor comprised a single CPU constructed out of 27 tons of discrete components that occupied an entire room – a scarce resource that had to be re-used to (sequentially) perform all the computations of a given application. In such a case, re-use would tend towards 100% and the machine would be relatively slow. On the other hand, if quadrillions of transistors are available to implement an application, individual transistors can be used to perform all the calculations in parallel. In such a case, re-use would tend towards 0% and the machine would be relatively fast.

It would also appear that re-use may be following some sort of cycle driven by the maturity of a given technology. During the first half of the 19th century the basic computer was a mathematician with a slide rule using decimal arithmetic (because man has ten fingers). To speed things up large numbers of mathematicians could be used to perform calculations in parallel, *Lewis Fry Richardson's* forecast factory being a good example [71]. In 1950 the ENIAC computer was developed by the University of Pennsylvania's Moore School of Electrical Engineering [103] and the use of teams of mathematicians suddenly ceased. The world adopted digital technology which is based on binary arithmetic (because a transistor can be either ON or OFF). Today contemporary electronics routinely employ quadrillions of transistors in parallel, but the re-use pattern is about to reset as the world starts to usher the qubit age.

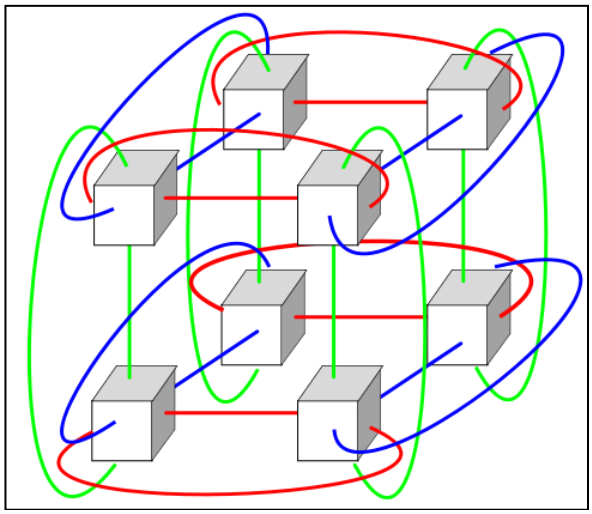


Figure 135. Three-Dimensional Torus Network used by Blue Gene Supercomputer [Wikipedia, 2018]

For the moment the world is moving towards a compromise solution somewhere between the centralization of CPU's and the decentralization of GPU's as described by *Fatahalian* and *Houston* [89]. For similar reasons the optimum mix for implementing the Array Processor is probably located somewhere between CPU's and FPGA's. While this mix is obviously influenced by the availability of quadrillions of transistors in contemporary electronics, it is also affected by other factors such as the spatial relationship implied by the circuit and the fact that

contemporary electronics are slices of Silicon contained in flat packages. The problem is illustrated in Figure 135 – how to connect thousands of small areas on slices of Silicon in a 3-dimensional grid with nearest neighbour interfaces as illustrated in the figure. In the case of a supercomputer like Blue Gene the cube of Figure 135 can be morphed into almost any shape because the need to limit space and length of nearest neighbour interfaces are not as strong a driver as in the case the Precision Rocket Launcher.

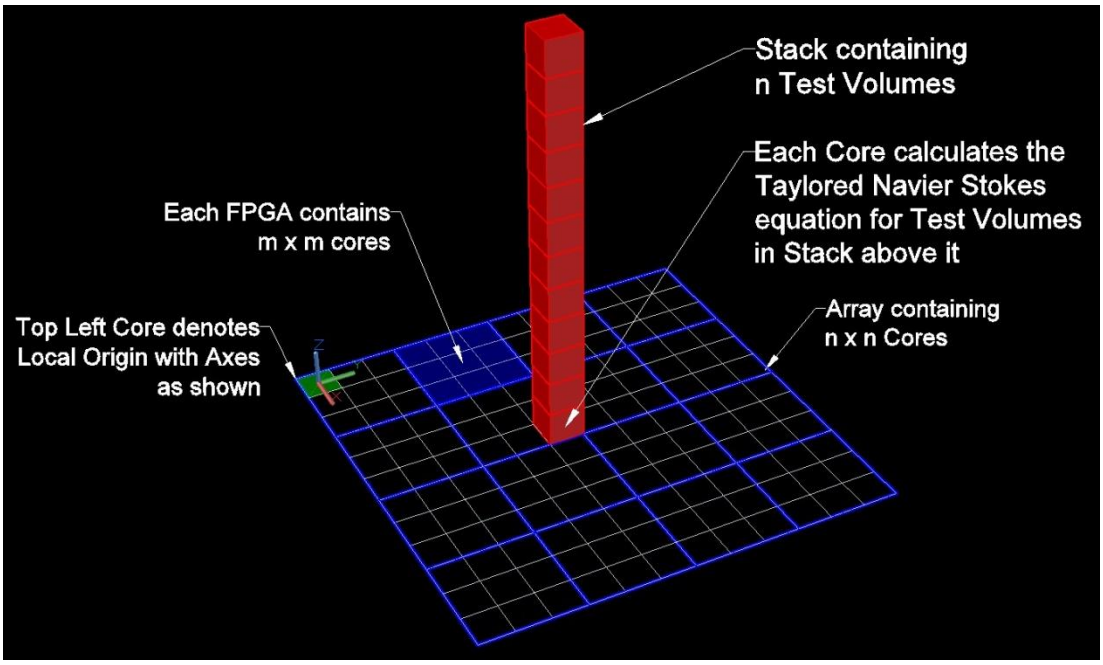


Figure 136. Functional Allocation of Array Processor [A Landman: 2017]

The compromise solution chosen for the Array Processor and tested for the purposes of this study is to implement two dimensions of the array in hardware (i.e. a low re-use level) and one dimension in firmware (i.e. a high re-use level). This yields a planar array of devices (i.e. a sheet) that can be rolled around the exterior of the M159 Rocket Launcher. The principle is illustrated in Figure 136 which shows an example of a small Array Processor comprising 144 cores embedded in a 4x4 array of FPGA devices with each FPGA device containing a 3x3 array of cores. Such an Array Processor would be representative of a Flight Box containing 12x12x12 test volumes with each core responsible for repeatedly calculating the Taylored Navier Stokes equation for a column or stack of 12 test volumes corresponding with the XY positions of the cores in the array.

As will be shown in Chapters 6 and 7 the cores all work in unison in this arrangement, simultaneously calculating the Taylored Navier Stokes equation for all the test volumes of a given layer in the Flight Box in a **minor cycle**. These minor cycles are sequentially repeated for all the layers from bottom to top, resulting in an estimate of the downwash flow field after completion of the resulting **major cycle**. Provided the Taylored Navier Stokes equation is implemented in deterministic fashion (this topic is re-visited in Chapters 6 and 7) these estimates occur at constant real time intervals. Note that the cores run in unison to enable each core to transmit its current state to its nearest neighbours. These exchanges enable each core to calculate the spatial derivatives of the flow field as described in step (d) of paragraph 4.5.4. It also enables expansion of the Array Processor by adding more FPGA's, either as a result of a need to enlarge the Flight Box or as a result of a need to reduce the grid size.

Note that the scheme as illustrated in Figure 136 implies that the array employs dedicated use (hardware) along the X and Y axes while the Z axis employs re-use (i.e. the same circuit used repeatedly for different inputs). The effect of this choice is that in physical terms the array is embedded in a thin sheet of FPGA's that can be wrapped around the exterior of the space tube to facilitate cooling with ambient air. Should each cell be implemented with a separate core, the sheet would be thick, making packing and cooling of the array very problematic.

Finally, it must also be noted that the scheme as illustrated in Figure 136 makes a fundamental assumption – that whichever FPGA device is chosen would be fast enough to complete the major cycle within 0,1 millisecond. Testing this hypothesis is the subject of Chapters 6 and 7.

5.5.2 The Commutation process

The mechanism as described above represents a scheme for uninterrupted implementation of the **Gas Flow Model** in real time using physical devices and Hardware Development Language (HDL). In its current form the model estimates the response of the **Space** block as illustrated in Figure 20 with interfaces to objects such as the main rotor, fuselage and ground absent. For such a system operating in complete isolation the flow field solution can at best be influenced by setting the initial conditions of the Array.

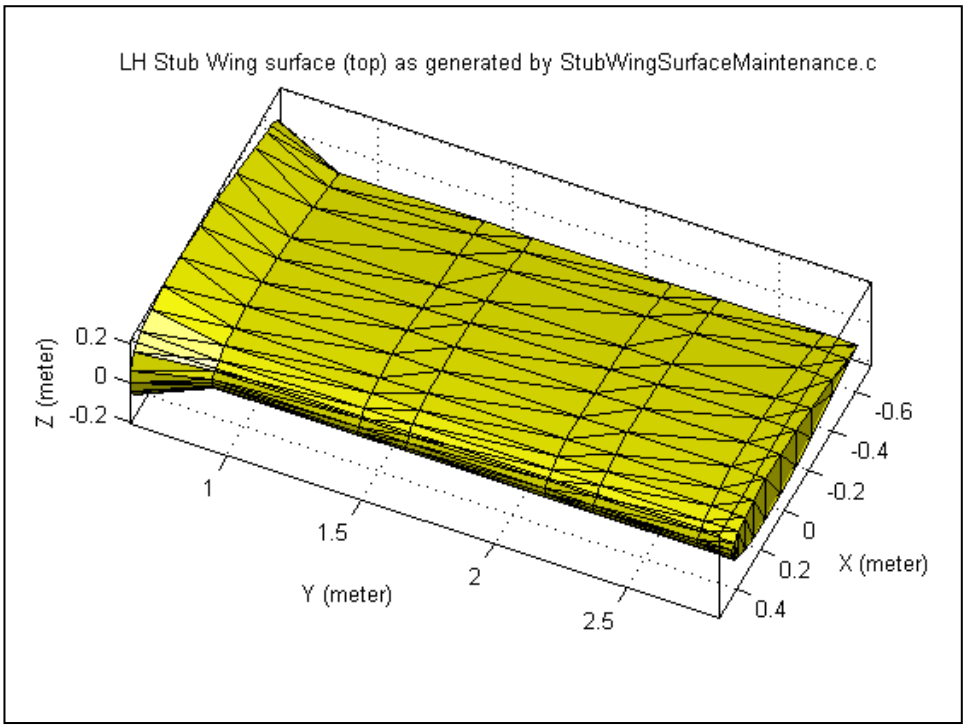


Figure 137. Shape Matrix of LH Stub Wing visualized [A Landman, 2014]

To introduce into the flow field solution the effects of bodies such as the main rotor, fuselage and ground as well as gas flows such as engine exhausts and wind use is made of a **Commutation** process through which the influence of external shapes of bodies within the Flight Box are imposed on the **Gas Flow** process. This can be achieved by freezing/unfreezing the internal states of successive sets of cells as function of the contents of a number of shape matrixes as generated external models such as the **Main Rotor** block of Figure 20. An example is illustrated in Figure 137 which is a graphical representation of the shape matrix of the LH Stub Wing of Rooivalk AH-2A as generated by the **LH Stub Wing** block of Figure 20.

The **Array Coordinator** block of the Array Processor as illustrated in Figure 134 will be used to determine those test volumes that are intersected by any of the surface patches contained in the Shape Matrix. Boundary conditions as supplied by the particular block are then imposed on the intersected test volumes. Provided the outputs of these models vary at frequencies well below the iteration frequency of the **Gas Flow** process the combined response of such a system should represent the dynamic behaviour of any combination of air and multiple objects such main rotor blades, fuselage, ground as well as sources of gas flow such as engine exhausts, rocket exhausts and ambient wind.

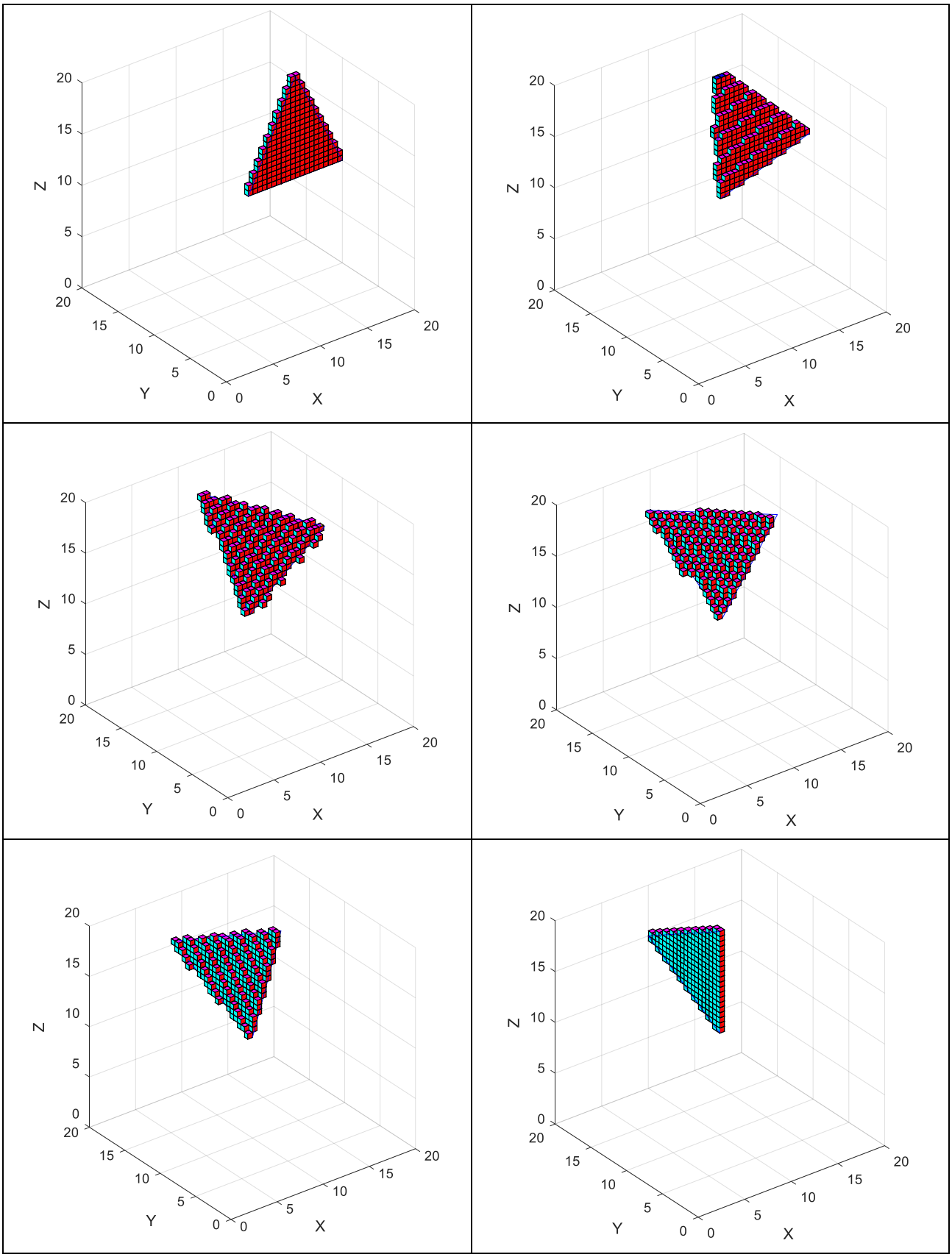


Figure 138. Example of rotating surface patch intersecting test volumes [A Landman, 2017]

Note: *The **Commutation** process is well suited for parallelization using a similar process used by a GPU to build a virtual scene. Researching this process was not the objective of this study and hence, was not studied in much detail. Only basic implementations of*

this function are implemented in Chapters 6 and 7. A detailed investigation of this process will have to be the subject of a future study as described in Chapter 8.

The **Commutation** process was emulated with a MATLAB program in which a single surface patch was rotated through the meshgrid of a small Flight Box. The result was a video sequence that illustrates the commutation between consecutive sets of test volumes as determined by the evolution in position and rotation of the surface patch between the time steps of the simulation. Some of the resulting video frames are illustrated in Figure 138.

5.5.3 The Probing process

The **Probing** process is used to interrogate the flow field solution at a pre-defined list of coordinates (in FPGA Axes) written into dual-port Random Access Memory (RAM) by an external block via the Modelling Bus as illustrated in Figure 134. The probing process is performed by a state machine located in the **Array Coordinator** block of the Array Processor which sequentially reads the solution at the specified coordinates and writes the results into dual-port RAM where it can be accessed by an external block via the Modelling Bus.

Note: *The Probing State Machine is essentially a pseudo-processor that executes a "program" comprising a set of rudimentary instructions. We will return to this aspect in Chapters 6 and 7.*

Interrogation of the solution is performed at times when the **Gas Flow** process is inactive (i.e. a time-slicing scheme). In addition, decoding of the Probe Address is done in hardware (with a network of logical gates) so that the Probing Rate is independent of the size of the Array Processor.

5.5.4 The Grid Hop Process

The **Grid Hop** process is used to "push" the Flight Box through Inertial Space in sympathy with the movement of the Aircraft. Stated differently, the **Grid Hop** process is used to move the Flight Box in such a way that the Navier Stokes solution is calculated for that part of space where the disturbance caused by Main Rotor, Tail Rotor, Fuselage and Engines is at it greatest.

Figure 139 is a graphical representation of the **Grid Hop** process, showing what happens during a Left-to-Right data shift. As shown, the internal state variables (p, T, u, v and w) of each cell is transferred to the adjacent right hand cell. All of these transfers happens simultaneously upon command issued by a state controller located within the **Array Coordinator** block. Upon completion of the mass data shift, the left boundary cells are over-written with ambient conditions as determined by an **Air Data Computation** block.

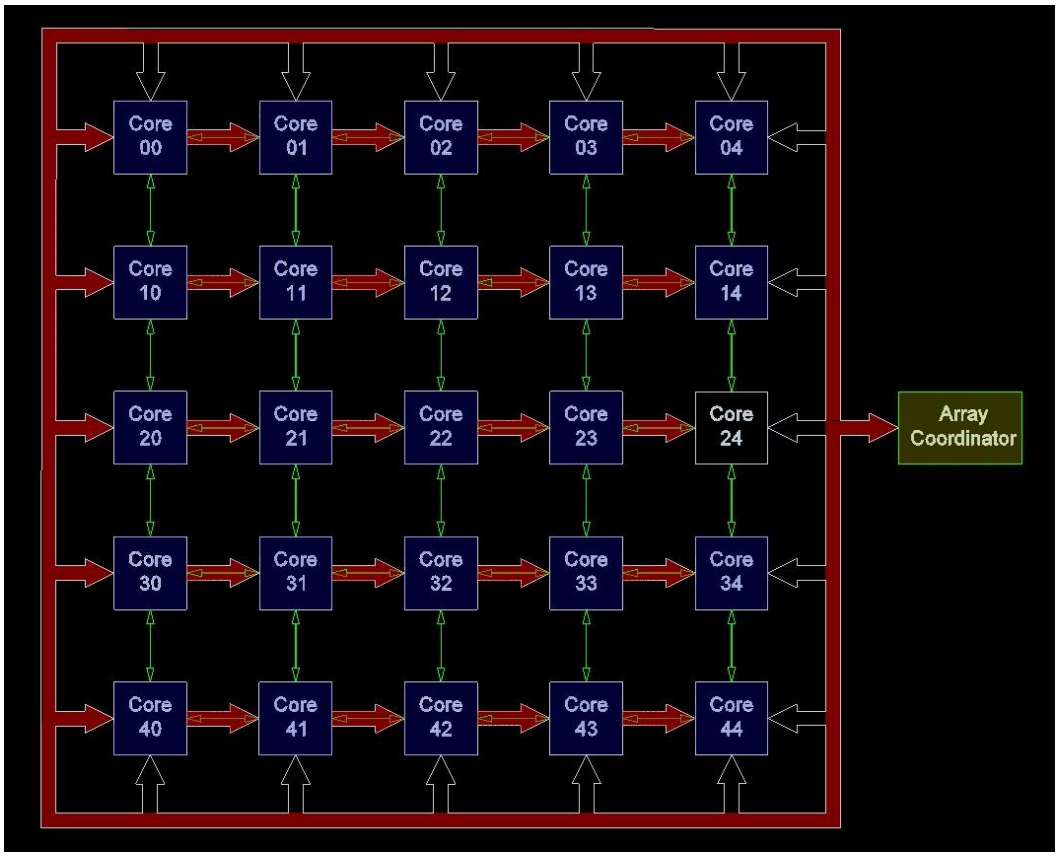


Figure 139. Data movement during Left-to-Right shift [A Landman, 2017]

The effect of the **Grid Hop** process as described above is that the "focus" of the **Downwash Model** is shifted with one grid spacing (200 mm in case of the PRL) per hop. Since the process is repeated once after every major cycle of the **Downwash Model**, the focus can move at a maximum speed of 0,2x10000 = 2000 meter/sec. This is more than fast enough to track the movement of the Aircraft through a continuous stream of shift operations such as Left-Right, Front-back and Up-down.

5.5.5 Sampling rate considerations

The **Gas Flow** process as described above takes discrete steps in both the temporal and spatial domains. As a result, the limitations of *Nyquist* [19] as described in Chapter 4 are applicable.

a) The grid spacing (i.e. spatial sampling rate) of the **Gas Flow** process must be less than 121 mm as derived in paragraph 4.5. Assuming a grid spacing of 200 mm and a Flight Box of 100m x 100m x 100m this implies an array of 500x500x500 or 125 million cells.

b) The time increment (i.e. temporal sampling rate) of the **Gas Flow** process must be less than 0,18 msec as derived in paragraph 4.5. Choosing an over-sampling factor of 2 sets the required time increment at 0,1 msec. Hence, for the **Gas Flow** process to estimate the downwash as it evolves in actual time, the model must preferably run at an iteration frequency of 10 kHz.

5.5.6 Packaging considerations

Apart from the need to carry and launch a number of FZ90 Rockets as illustrated in Figure 121 the Precision Rocket Launcher must also serve as a container that protects and holds a large number of FPGA's in a given spatial relationship. The optimum spatial relationship is obvious – the FPGA's should be located in a thin layer on the longitudinal periphery of the Precision Rocket Launcher where the thermal paths to ambient air would be the shortest and the FPGA's won't interfere with the trajectories or backblast flow fields of the FZ90 Rockets.

Assuming the FPGA's to be contained in a "shell" along the longitudinal periphery of the Precision Rocket Launcher, the best solution would be achieved if the FPGA's can be arranged in a "sheet" rolled into a cylinder forming the outer surface of the launcher. This will enable direct cooling of the FPGA's by channeling their heat directly to ambient air via suitable heat sinks.

5.5.7 Aircraft Constraints

In Rooivalk AH-2A the ballistic calculations for rockets are performed by the Rocket Launchers, each of which is comprised of a conventional M159 Rocket Launcher to which a Rocket Electronic Unit (REU) is permanently attached as illustrated in Figure 121. This is the arrangement as originally prescribed by the *Author* [104] to provide intelligent weapon stations on Rooivalk AH-2A in accordance with the requirements of MIL-STD-1760A [102]. This standard is the North Atlantic Treaty Organization (NATO) equivalent of a "plug and play" weapon where the peculiarities of a given weapon reside within the weapon itself (usually the launcher) and the aircraft acts as a general platform.

The specific functionality required on the aircraft to operate a given weapon is "installed" on the aircraft and run when the weapon is attached to one of the weapon stations of the aircraft. Since both the M159 Rocket Launcher and FZ90 Rocket as used on Rooivalk AH-2A are dumb devices, the intelligence for the weapon stations as called for by MIL-STD-1760A [102] is supplied by the REU's. The REU's are also responsible for generating firing pulses and implementing safety interlocks and discrete signals such as *Release Consent* as prescribed by MIL-STD-1760A [102]. In formulating a conceptual design for the PRL the same principles were followed here albeit with a few exceptions. For example, whereas the REU and M159 Rocket Launcher were separate LRU's on Rooivalk AH-2A, they are now integrated into a single LRU.

5.5.8 Main Electronics Unit

The Main Electronics Unit (MEU) covers the top surface of the space tube of the Precision Rocket Launcher. It will contain a Host Processor (HP), Digital Signal Processors (DSP) and other electronics necessary for combining dual power feeds from the Aircraft, generation of regulated electrical power to drive the PRL, communication with the Aircraft in accordance with the MIL-STD-1760A [102] protocol, controlling the Array Processor, performing ballistic calculations and controlling the FZ90 Rockets and their Electro-Magnetic Launch Tubes. This Shop Replaceable Unit (SRU) will be fitted with a lanyard connector (jettison provision) to provide the electrical connections between the PRL and Aircraft as defined by MIL-STD-1760A [102].

Note: *For the purposes of this study we will only zoom into some of the functionality of the Array Controller and how it interacts with 250000 cores to form a Planar Array Processor that can be used to solve a Computation Fluid Dynamics problem in real time (10 kHz). Please note that the Array Controller will be located within the Main Electronics Unit while the cores will be located in 25 Array Processor Modules spread around the periphery of the PRL.*

5.5.9 Array Processor Module (APM)

The Array Processor Module (APM) will be comprised of a multi-layer Printed Circuit Board containing eight male connectors (facing downwards) as illustrated in Figure 140. These connectors will mate with eight corresponding female connectors on the Wrap-Around Printed Circuit Board when the Array Processor Module is inserted into the PRL.

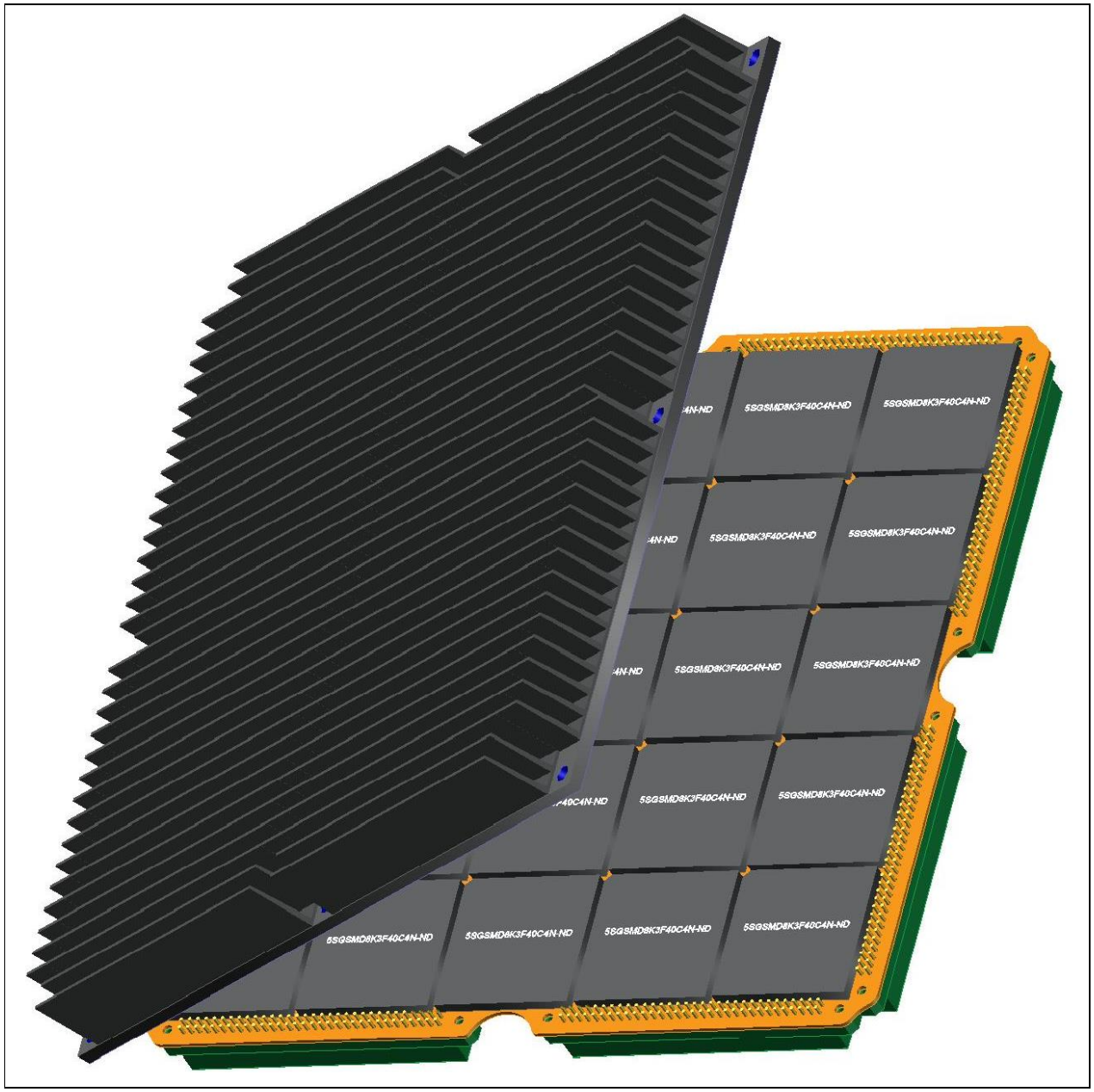


Figure 140. Array Processor Module with Heatsink removed [A Landman, 2017]

Mounted on the opposite side of the Printed Circuit Board will be a 5x5 array of Field Programmable Gate Arrays with their top surfaces clamped against a APM heat sink (to form a mechanical interface with low thermal resistance) as illustrated in Figure 140. The APM heat sink is characterized by an overall size of 200 x 200 mm, fin height of 25 mm, fin thickness of 2 mm and inter-fin gap of 8 mm.

Note: *Figure 140 illustrates the layout of the Array Processor Module when INTEL device 5SGSMD8K3F40C4N-ND is used. This is a 40 x 40 x 3,7 mm device that results in a 2mm gap between adjacent devices, yielding a 5 x 5 array of FPGA's as shown. The*

5SGSMD8K3F40C4N-ND is one of the top-end FPGA's of ALTERA known as the Stratix V series.

Since the PRL is mounted below the Stub Wing of Rooivalk AH-2A the APM heat sink will be exposed to ambient air, thus cooling the Field Programmable Gate Arrays below. The Field Programmable Gate Arrays will be Ball Grid Array (BGA) devices, mounted with "putty pads" as illustrated in Figure 141 and described by *Miraglia* [105] so as to ensure a low thermal resistance between FPGA tops and the APM heat sink.

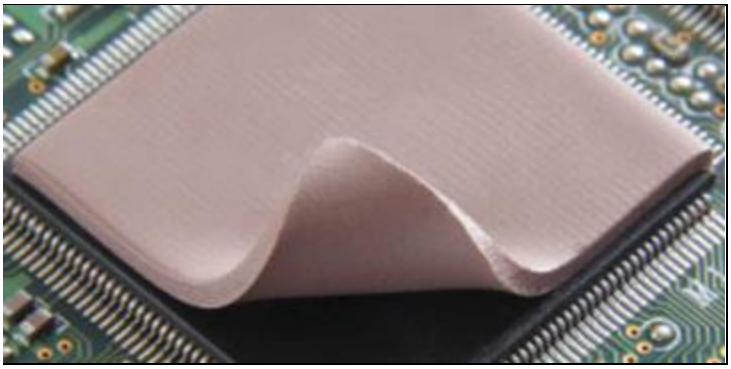


Figure 141. Putty pads providing low thermal resistance to heat sink [Miraglia, 2018]

In paragraph 5.2 the overall power dissipated by the Array Processor was shown to be 9,1 kW. This implies each Array Processor Module will dissipate 364 W which would have to be removed with the aircraft flying (forced convection with typical ambient wind speed of 40 m/s) or stationary during maintenance (natural convection with ambient wind stationary). It is clear that from a thermal perspective the latter case would be the worst.

Note: *Each Array Processor Module is assumed to dissipate 124 Watt (versus 364 Watt actual) henceforth. This figure stems from a previous design cycle, but the principles used and conclusions drawn remain the same.*

As described by *Coulson* and *Richardson* [39] the heat transfer from a body to an ambient fluid through the process of convection is dominated by a thin layer of fluid covering the body. The heat transfer through this layer is given by the following equation:

$$Q = hA\left(T_1 - T_2\right) \qquad \textbf{Equation 114}$$

With Q the overall heat transferred through the convection layer, h the heat transfer coefficient of the convection layer, A the surface area of the body, T_1 the temperature of the body and T_2 the temperature of ambient fluid. Irrespective of the apparent simplicity of Equation 114 convection is a complex process that depends on the particular configuration at hand. To solve this problem *Coulson* and *Richardson* [39] use dimensional analysis to estimate the heat q transferred from a unit area of a body in unit time to an ambient fluid through the process of convection. Their analysis yields the following equation:

$$\frac{qL}{k\Delta T} = \frac{hL}{k} = C\left(\frac{Lu\rho}{\mu}\right)^{W1}\left(\frac{C_p\mu}{k}\right)^{W2}\left(\frac{\beta g\Delta TL^3\rho^2}{\mu^2}\right)^{W3} \qquad \textbf{Equation 115}$$

with u : Velocity of ambient fluid relative to body

L : Characteristic dimension of object

μ : Viscosity of ambient fluid

ρ : Density of ambient fluid

k : Thermal conductivity of ambient fluid

C_p : Specific heat at constant pressure of ambient fluid

ΔT : Temperature difference between body and ambient fluid

β : Coefficient of thermal expansion

g : Gravitational acceleration

C : Constant to be determined through empirical means

$W1$: Constant to be determined through empirical means

$W2$: Constant to be determined through empirical means

$W3$: Constant to be determined through empirical means

The $\left(\frac{hL}{k}\right)$ term is the Nusselt group N_u, the $\left(\frac{Lu\rho}{\mu}\right)$ term is the Reynolds group R_e, the $\left(\frac{C_p\mu}{k}\right)$ term is the Prandtl group P_r and the $\left(\frac{\beta g\Delta TL^3\rho^2}{\mu^2}\right)$ term is the Grashof group G_r. If the heat transfer is dominated by natural convection the velocity of the ambient fluid is small and the Reynolds group can be omitted. If the heat transfer is dominated by forced convection the effect of natural convection is small and the Grashof group can be omitted. Although the Prandtl group is applicable to both cases it is constant for air over a wide temperature and pressure range, thus simplifying determination of the heat transfer coefficient of objects in air.

5.5.9.1 Cooling the APM with Natural Convection

Applying the principles as described in paragraph 5.5.9 *Coulson* and *Richardson* [39] show that in the case of natural convection Equation 115 reduces to the following:

$$Nu = C^{'}\left(Pr.Gr\right)^{n} \qquad \textbf{Equation 116}$$

with C' a constant that has a value of 0,56 for vertical planes or 0,54 for horizontal planes facing upward or 0,25 for horizontal planes facing downward. Parameter *n* has a value of 0,25 for streamline flow and 0,33 for turbulent flow. In the case of air Equation 116 can be simplified even further, resulting in the equations as listed in Table 6 with the heat transfer coefficients in units of lb.cal/ft^2.hr.°C.

Table 6. Heat transfer coefficients for plates in air [Coulson and Richardson, 1970]

Configuration	Streamline	Turbulent
Vertical planes	$h = 0{,}32\left(\frac{\Delta T}{l}\right)^{0{,}25}$	$h = 0{,}35\left(\Delta T\right)^{0{,}25}$
Horizontal planes facing upwards	$h = 0{,}31\left(\frac{\Delta T}{l}\right)^{0{,}25}$	$h = 0{,}41\left(\Delta T\right)^{0{,}25}$
Horizontal planes facing downwards	$h = 0{,}14\left(\frac{\Delta T}{l}\right)^{0{,}25}$	Turbulent flow not reached in this configuration

What is evident from Table 6 are two facts. First, the heat transfer from a vertical plane is about three times more than that of a horizontal plane facing downwards and second, heat transfer is proportional to $(\Delta T)^{0,25}$ which allows extrapolation from a known work point (we will return to both of these facts shorly).

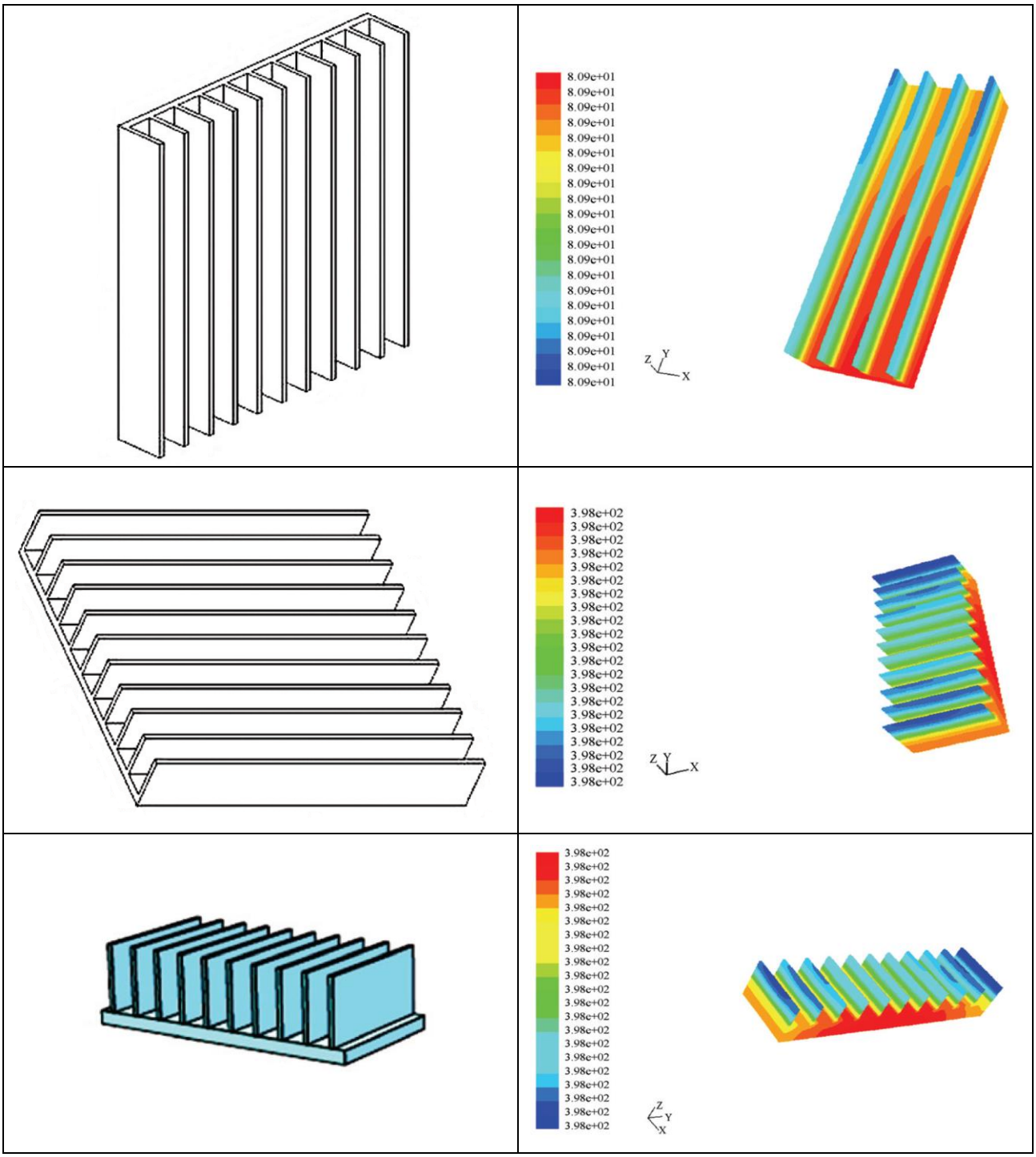


Figure 142. Vertical/vertical, vertical/horizontal and horizontal/up heatsinks [Goshayeshi et al, 2009]

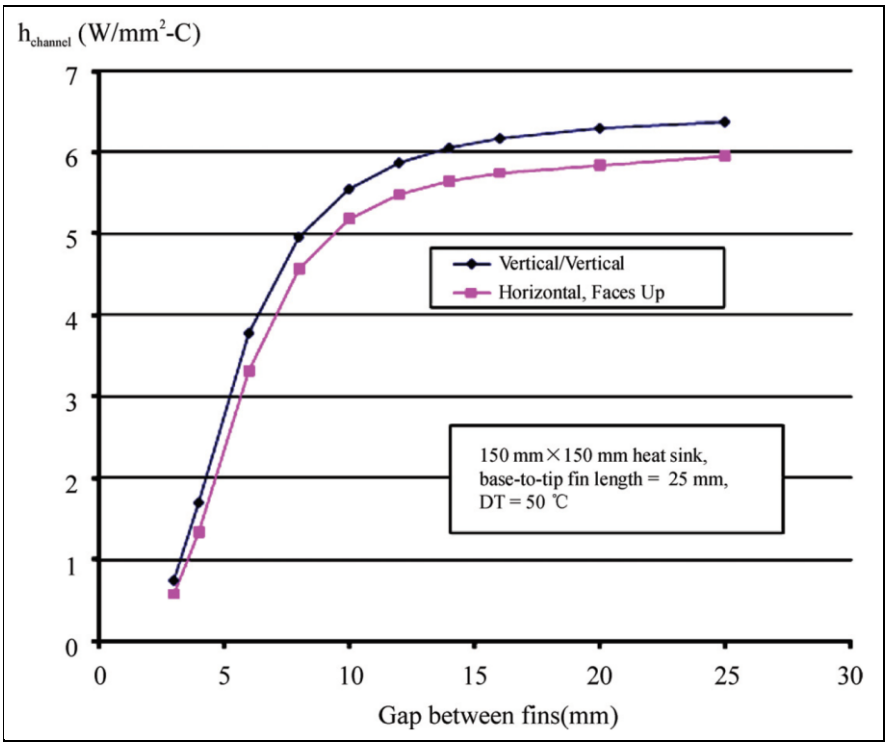


Figure 143. Heat transfer coefficient of a channel from 150x150 mm vertical/vertical and horizontal/up heatsinks versus inter-fin gap [Goshayeshi et al, 2009]

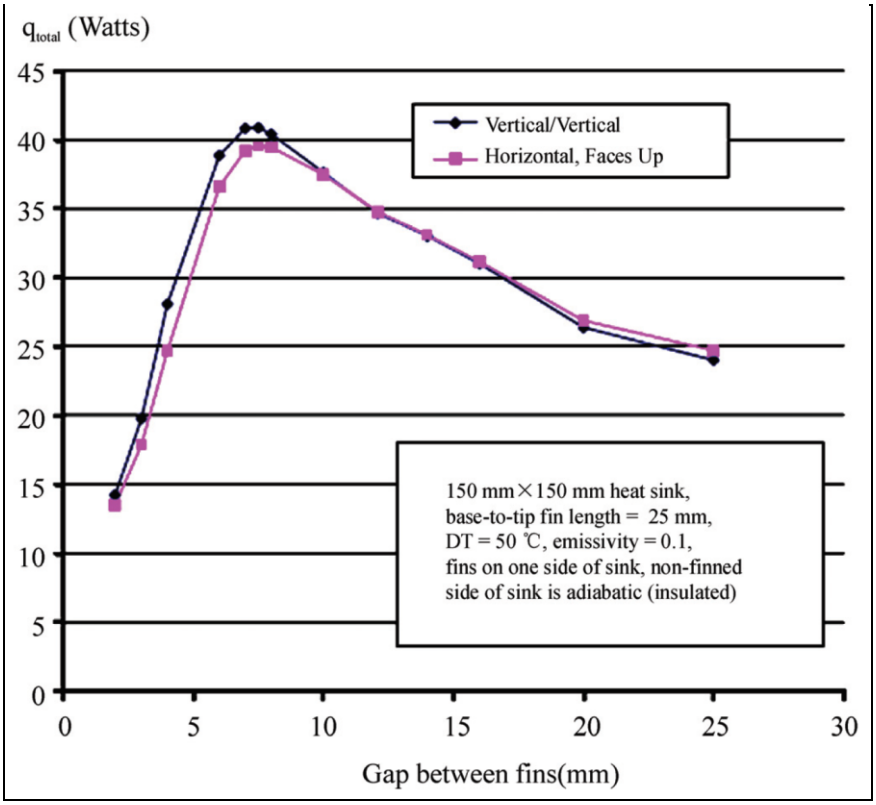


Figure 144. Total power flows from 150x150 mm vertical/vertical and horizontal/up heatsinks versus inter-fin gap [Goshayeshi et al, 2009]

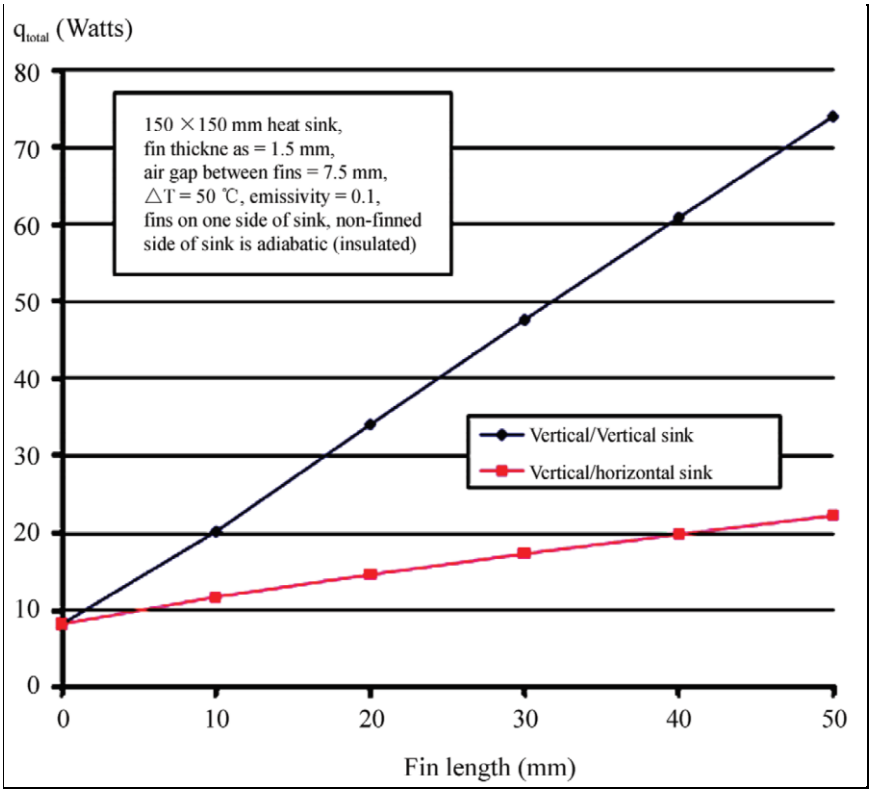


Figure 145. Total power flows from 150x150 mm vertical/vertical and vertical/horizontal heatsinks versus fin height [Goshayeshi et al, 2009]

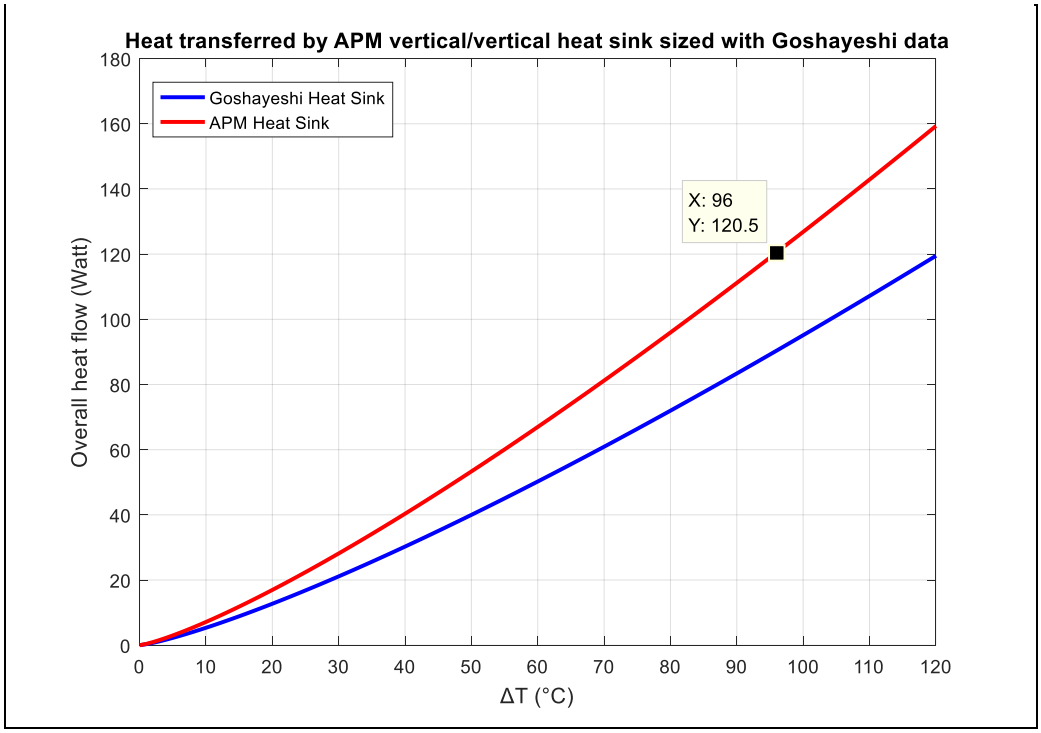


Figure 146. Heat shed by APM heatsink with vertical/vertical configuration [A Landman, 2018]

The open literature contains a large body of material describing numerous experiments in which the constants of Equation 115 were determined for a wide range of scenarios (of which only a few are applicable here). One of these studies was conducted by *Goshayeshi* and *Ampofo* [106] who used the FLUENT program to calculate the heat transferred from finned heatsinks in vertical

and horizontal orientations with fins aligned in various directions. Analyzing the vertical/vertical and horizontal/up configurations as illustrated in Figure 142 for a 150 x 150 mm heatsink with temperature 50°C above ambient they found the results as illustrated in Figure 143 and Figure 144 for different inter-fin gaps. It is evident that the heat shedded by such a device peaks at 40 Watt with an inter-fin gap of about 7 mm for both configurations because both result in similar air flows.

Note: *FLUENT is a Computational Fluid Dynamics program produced by ANSYS of Canonsburg, Pennsylvania in the USA.*

If the airflow for such a device would scale linearly an APM heat sink with vertical/vertical configuration would be able to shed about 53 Watt at the same temperature difference. In reality the longer fins of an APM heat sink will cause a reduction in airflow which would have to be countered by increasing the inter-fin gap, resulting in a different power peak. Ultimately an APM heat sink with vertical/vertical configuration would probably shed slightly less power – about 50 Watt. In contrast it follows From Table 6 that an APM heat sink with fins facing downwards (as is the case at the bottom of the PRL) will only be able to shed about 16,7 Watt with heat sink temperature 50°C above ambient.

Performing a similar analysis for different fin heights of a vertical/vertical heat sink and a vertical/horizontal heat sink *Goshayeshi* and *Ampofo* [106] found the result as illustrated in Figure 145 from which it is evident that a vertical/vertical configuration can shed three times as much power as a vertical/horizontal configuration. Hence, assuming linear scaling an APM heat sink with vertical/horizontal configuration would only be able to shed about 22,5 Watt with an inter-fin gap of 7,5 mm and fin height of 25 mm.

It is clear that an APM heat sink with vertical/vertical configuration at a temperature of 50°C above ambient air (as assumed by *Goshayeshi* and *Ampofo*) would not be able to remove the heat generated by the APM. For an APM mounted at the bottom of the PRL this problem would be even worse.

Assuming that the fin height of the APM heat sink cannot be increased beyond 25 mm (because it will make the PRL overly fragile during handling) the only alternative for removing the 120 Watt of heat generated through natural convection would be to increase the temperature difference between heat sink and ambient air. To estimate the temperature difference where this would occur it is evident from Table 6 that the heat transfer coefficient *h* for natural convection is proportional to $\Delta T^{0,25}$ for all heat sink configurations. Combining this fact with Equation 114 means the power shedded by a heat sink through natural convection is proportional to $\Delta T^{1,25}$ for all heat sink configurations. Under this assumption the power transfer curve of the 150x150 mm vertical/vertical heat sink of *Goshayeshi* and *Ampofo* [106] plotted against temperature difference is as illustrated by the blue curve in Figure 146.

Assuming that linear scaling would hold for relatively small changes in heat sink size, the power transfer curve of an APM heat sink with vertical/vertical configuration should be as illustrated by the red curve of Figure 146 indicating that a temperature difference of about 95°C would be required to shed 120 Watt. For an ambient air temperature of +40°C this would require a heat sink temperature of 135°C. A temperature difference of about 10°C between PFGA core and FPGA surface would be desirable based on practical experience. Even so the FPGA core would be operating at 145°C which is way above the absolute maximum junction temperature (125°C) of the STRATIX 10 device [107]. For an APM located at bottom of the PRL this temperature problem would be even worse.

5.5.9.2 Cooling the APM with Forced Convection

It is evident from the analysis above that natural convection by itself would be insufficient for cooling the Precision Rocket Launcher and that some other mechanism would be required instead. One approach as investigated by *Khabari et al* [108] is to sputter copper onto the

surface of a heat sink, thus increasing the overall surface area of the heat sink due to the uneven nature of the nanoparticle layer thus created. Depositing such a layer on the clip-on heatsink of a small silicon transistor (2N2222A) they found a 4,0°C reduction in case temperature under conditions of natural convection. Under conditions of forced convection the reduction was smaller - 1,5°C at a wind speed of 2,9 m/s and 0,5°C at a wind speed of 3,7 m/s.

It is evident that the coating as developed by *Khabari et al* [108] would help but does not constitute a solution to the problem considered here. Although undesirable from a complexity point of view the only viable solution for achieving a reasonably low case temperature for the APM heat sink appears to be forced convection. Applying the principles as described in paragraph 5.5.9 to the forced convection case under conditions of laminar flow *Bahrami* [109] shows that Equation 115 reduces to the following:

$$Nu = 0,332 Re^{0,5} Pr^{0,3333}$$ **Equation 117**

where a Reynolds number less than 5x10^5 designates laminar flow. In the case of turbulent flow *Bahrami* [109] shows that Equation 115 reduces to the following:

$$Nu = 0,0296 Re^{0,8} Pr^{0,3333}$$ **Equation 118**

For air the Prandtl group is essentially constant over the velocity and temperature range applicable to the APM heatsink so that the heat transfer coefficient *h* becomes directly proportional to $Re^{0,5}$ for laminar flow and directly proportional to $Re^{0,8}$ for turbulent flow.

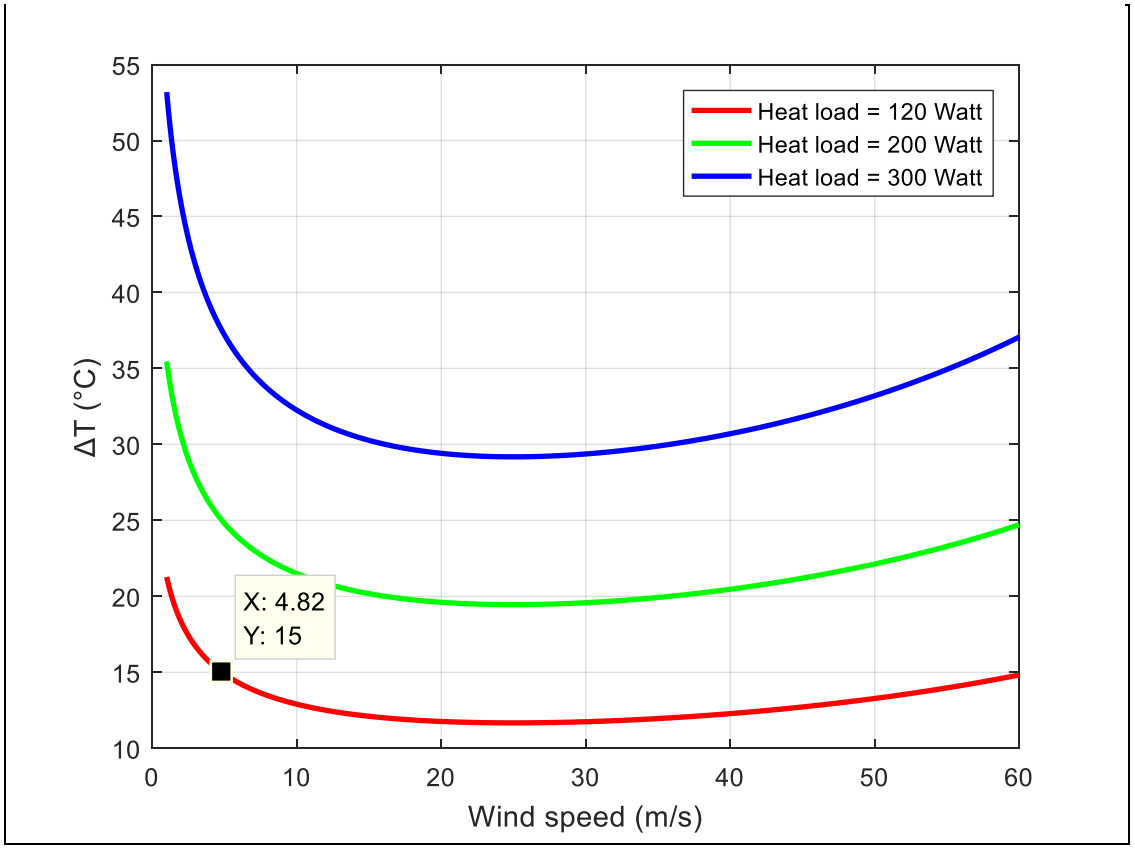


Figure 147. Temperature difference versus wind speed: APM heat sink [A Landman, 2018]

As described by *Osczevski* [110] the heat transfer coefficient of a body under conditions of laminar flow was determined experimentally by *Siple* and *Passel* during an expedition to

Antarctica in 1941. Their experiment involved exposing a small plastic bottle (2,5 inch in diameter containing water) to the arctic environment and measuring the time taken for the contents to freeze under various wind speeds. Knowing the heat capacity of water they were able to formulate the following relationship:

$$h = 10,45 - u + 10u^{0,5} \qquad \textbf{Equation 119}$$

with *h* the heat transfer coefficient for forced convection under conditions of laminar flow in units of Watt/m^2.°K. Given the fact that for air the Prandtl group is essentially constant under conditions applicable here and noting that $R_e = \left(\frac{Lu\rho}{\mu}\right)$ the correspondence between Equation 117 and Equation 119 is evident.

Note: *The work performed by Siple and Passel [110] in deriving Equation 119 was an attempt to quantify the process of frost bite in humans. To this effect their water bottle was supposed to model the thermal characteristics of a human face. This solicited various criticisms as described by Osczevski [110], one being that the thermal resistance of the bottle wall was not suitably considered. Considering the plastic that the bottle was made off, Osczevski estimates the thermal resistance of the bottle wall in the range 0,014 to 0,024 m^2°K/W and concludes that it had to be three to five times thicker to emulate the human face. In the case of a metal heat sink this thermal resistance will effectively be absent, so Equation 119 will yield a slight underestimation of the heat transfer coefficient of a metallic body subjected to forced convection. This is acceptable for the purposes of this study since it will result in a slight over-design.*

Assuming the top of the APM heat sink to be covered with an insulating layer to avoid the PRL from becoming a target for man-portable air-defense systems (MANPADS) the total surface area of the APM heat sink exposed to airflow will be 25*2(0,025+0,004)*0,2 = 0,29 m^2. Combining this fact with Equation 114 and Equation 119 the wind speed required to dispose of 120 Watt, 180 Watt and 240 Watt of heat through the APM heat sink will as illustrated in Figure 147. Note that the curves reach a minimum at 25 m/s. In reality the curves will reach (and stabilize) at even lower temperature differences since the heat transfer coefficient of a duct is lower for turbulent flow than laminar flow as described by *Coulson* and *Richardson* [39]. This behaviour is due to the form of Equation 119 which only holds for laminar flow. Assuming a length of 200 mm as characteristic dimension the Reynolds number *Re* for airflow within the APM heat sink is given by the following equation:

$$\begin{aligned} R_e &= \frac{0,2\,u\,1,2929}{1,861.10^{-5}} \\ &= 0,1389.10^5 u \end{aligned} \qquad \textbf{Equation 120}$$

Based on Equation 117 and Equation 120 it is evident that the airflow within the APM heat sink will remain laminar up to a speed of 36 m/s. This is in reasonable agreement with the minimum at 25 m/s as illustrated in Figure 147.

5.5.9.3 Maintaining a low FPGA core temperature

It is evident from the analysis above that 120 Watt can be shedded with ease when forced convection is employed in the APM heat sink. A suitable workpoint at a temperature difference of 15°C is illustrated in Figure 147. Establishing a wind speed of 4,82 m/s through the APM heat sink will require a flow rate of 3,47 m^3/hr which will be relatively easy to accomplish with axial fans that have much larger maximum flow rates (depending drag of duct containing the flow).

Establishing the required flow rate is not a problem and hence, will not be pursued any further in this study.

Assuming a maximum ambient temperature of 40°C and the workpoint as illustrated in Figure 147 the maximum temperature of the APM heat sink will be 55°C. This means the FPGA will run at a core temperature of 65°C assuming a typical temperature difference of 10°C between core and package case. This is well within the recommended maximum junction temperature of 100°C for the STRATIX 10 device family [107].

Although the maximum FPGA core temperature will not exceed 65°C assuming the scenario as described above it may still be too high from a reliability point of view. This may be a problem in the case of the Precision Rocket Launcher because the Array Processor will comprise 625 separate FPGA devices with high temperature being a root cause of low Mean Time Between Failures (MTBF). Reliability considerations may therefore call for a mechanism that would ensure each FPGA runs at a relatively low junction temperature such as 40°C.

What would be required to achieve this is on the PRL is an active cooling mechanism that is robust, easy to implement, requires little space, has a high Mean Time Between Failures, requires little maintenance and is certain to work under both ground-bourne and airborne conditions. *Brown et al* [111] describes various contemporary mechanisms for cooling electronic equipment including Magnetic Cooling, Thermionic Cooling, Thermoacoustic Cooling, Thermoelectric Cooling and Thermotunneling Cooling. Although some of these mechanisms hold great promise, most of them are still being researched or require an elaborate installation that would be unsuitable for the Precision Rocket Launcher. *Brown et al* [111] conclude that as at 2010 thermoelectric devices were the only mechanisms that have reached fruition with numerous commercially available components, in particular Peltier devices. This type of device is the only mechanism that complies with most of the requirements as listed above and hence, holds the most promise for cooling the Array Processor Module.

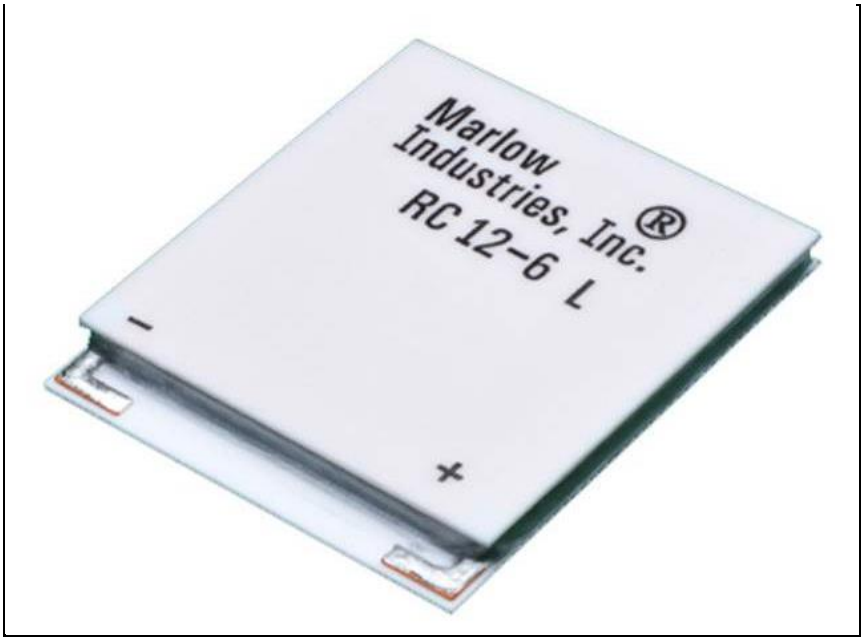


Figure 148. Physical outline of MARLOW Peltier device RC12-6L [MARLOW, 2018]

A typical single-stage Peltier device is illustrated in Figure 148. As shown it is comprised of a single layer of semiconductor pellets (usually Bismuth Telluride) sandwiched between two plates. The pellets are doped such as to yield alternate N and P materials that are electrically connected in series and thermally in parallel. Heat is transported from one side to the other side by minority carriers set in motion by passing a current through the device. While this current causes heat to be pumped through the device it also results in I^2R losses in the form of heat.

Hence, there exists a current where the heat pumped by a Peltier device is at a maximum as illustrated in Figure 149 for MARLOW device RC12-6L.

Note that each APM will contain a total of 25 STRATIX 10 FPGA's that will dissipate a total of 120 Watt, so each individual FPGA will dissipate 4,8 Watt. The size of STRATIX 10 device earmarked for use on the APM is about 42 x 42 x 3 mm while the size of MARLOW device RC12-8-01LS is about 44 x 40 x 4 mm. Hence, assume for the purpose of this study that each FPGA device will be cooled by a Peltier device RC12-8-01LS.

To ensure a junction temperature of 40°C the Peltier device would thus have to yield a differential temperature of 25°C at a heat load of 4,8 Watt. However, it is evident from Figure 149 that such a condition will require a Peltier current of about 1,25 Amp which will induce an additional heat load of 3,4 Watt because the internal resistance of MARLOW device RC12-8-01LS is 2,2 Ohm. This additional heat will increase the overall heat load of the APM heat sink to about 205 Watt. This additional heat load will cause a rise in APM heat sink temperature that can be countered by increasing the air flow through the APM heat sink. At best the required temperature differential ΔT of the APM heat sink temperature will now be 20°C with a wind velocity of 25 m/s. This rise in temperature will in turn require a slight rise in Peltier current, repeating the cycle until the system reaches an equilibrium of workpoints that will probably stabilize at a junction temperature of about 45°C and an overall heat load of about 210 Watt.

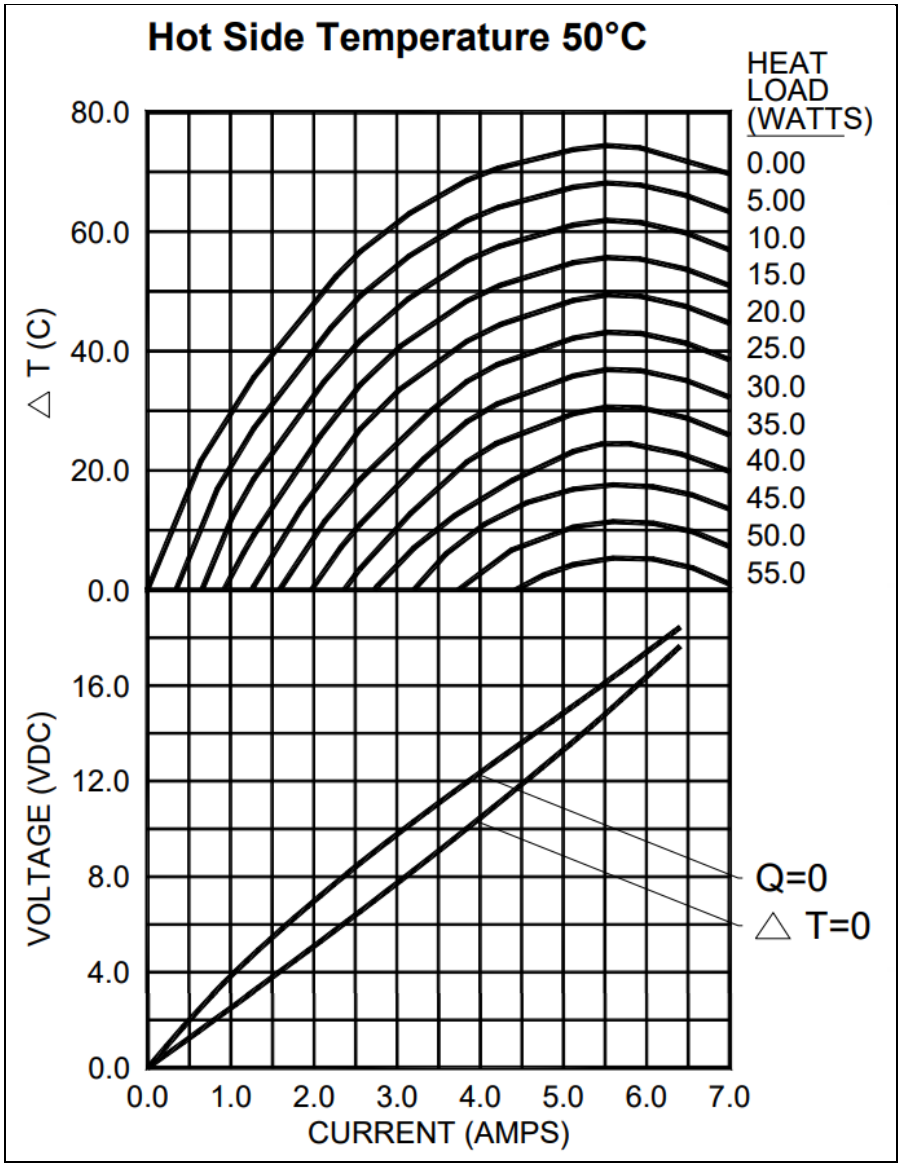


Figure 149. Characteristics of MARLOW Peltier device RC12-6L [MARLOW, 2018]

It is evident that using Peltier devices to reduce the maximum junction temperature of FPGA's used in the Array Procesor from 65°C to 45°C would double the overall power requirement of the Precision Rocket Launcher from 3,1 kW to about 6 kW. Acceptability of this fact in an actual design will depend on considerations such as mission reliability, mission profile and power budget at aircraft level.

5.6 Summary (Develop General Theories)

This chapter was used to perform the fifth Scientific Method Cycle of a family of nested Scientific Method Cycles that yield the concept of a Precision Rocket Launching System. Chapter 4 derived a **Gas Flow Model** that is suitable for solving the Navier Stokes equation for a given flight box and implementation with massive parallel schemes. The only boundary conditions required for the **Gas Flow Model** are those at the outer boundaries of the box. The first purpose of his chapter was to find mechanisms for controlling the **Gas Flow Model** in such a way as to model real world situations such as the downwash of a helicopter. The second purpose of his chapter was to find a hardware design of a Precision Rocket Launcher with structure and performance suitable to implement the **Gas Flow Model** combined with controlling mechanisms (refrerred to as the **Downwash Model**) in real time.

A wide range of topics such as the outline and dimensions of the M159 Rocket Launcher, internal architectures of Central Processing Units (CPU), Graphic Processing Units (GPU) and Field Programmable Gate Arrays (FPGA) as well as the apparent limitations posed by the Amdahl and Gustafson laws were presented as observational evidence.

Probing questions were asked as to what circuit and what type of device would represent the best approach for implementing the **Downwash Model** in real time. The answers to these questions were used to formulate the Automata Hypothesis which postulates that it should be possible to construct a combined data flow block diagram of the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes and to embed this diagram in an Array Processor comprised of a 2-dimensional array of FPGA devices rolled around the exterior of a Precision Rocket Launcher.

Some predictions based on the Automata Hypothesis were made, the main prediction being that it should be possible to package a 25x25 array of top-end FPGA's together with associated electronics into a Precision Rocket Launcher with throughput of at least 0,601 petaflop and size similar to the M159 Rocket Launcher currently used on Rooivalk AH-2A. A detailed design of the Precision Rocket Launcher was performed to test the predictions of the predictions of the Automata Hypothesis. This test proved the Automata Hypothesis correct, and we conclude that there is in existence at least one practical layout with dimensions similar to those of the M159 Rocket Launcher that can be implemented in contemporary electronics with absolute maximum throughput of 5,75 petaflop (an order of magnitude more than required) for executing the **Downwash Model** in real time.

This chapter made three contributions to the knowledge database:

- It layed a foundation for relating the performance of the Array Processor to the Amdahl and Gustafson laws and showed that the apparent limitation of Amdahl can be overcome by keeping the housekeeping function of such a processor constant. Implementing this conclusion in a real circuit therefore becomes the first objective of Chapters 6 and 7.

- It provided a process description of how the **Downwash Model** can be translated into a physical circuit – which becomes the second objective of Chapters 6 and 7.

- It provided a description of how a real Precision Rocket Launcher can be constructed, including the mechanical layout, electrical architecture and thermal design of such a device. It also showed how a Planar Array Processor can be created and embedded in such a system.

Note: *Apart from some qualitative descriptions in MIL-HDBK-762(MI) [29] regarding things to be considered during design of a Rocket Launcher no description of a Precision Rocket Launcher for an airborne application could be found in the open literature. The closest match found was work done by Wim van der Bauwhede et al of Glascow University [112] who managed to pack 1000 cores (each performing the Forward Discrete Cosine Transform) into a high-end FPGA device similar to STRATIX 10 as used here. Note however the difference in scale – one FPGA containing 1000 cores in case of the van der Bauwhede design versus 625 FPGA's containing 250000 cores (implementing 125 million cells) in this study.*

6 The Concurrent Model (Cycle 6)

"For a successful technology, reality must take precedence over public relations, for nature cannot be fooled". Richard P Feynman.

This chapter introduces the sixth Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. As shown in Chapter 2 the downwash of the helicopter must be calculated in real time for the PRLS concept to work. To this effect, Chapter 4 developed an algorithm which can be duplicated into a large number of cells that can be arranged in a 3-dimensional array called the **Gas Flow Model** which can be configured to calculate the flow field of any Flight Box to any desired accuracy. Chapter 5 conceived the **Downwash Model** (a combination of the **Gas Flow Model** and suitable mechanisms to control it) as well as an Array Processor comprising an array of FPGA's (part of Precision Rocket Launcher) with high enough throughput to implement such a system.

The main purpose of this chapter is to develop a circuit that implements the **Downwash Model** and which can be embedded with the resources and Planar Array Processor of the Precision Rocket Launcher as defined in Chapter 5. The study leads to a circuit referred to as the **Concurrent Model** – a solution with very high speed on the one hand but heavy resource requirements on the other.

Figure 150. Merrick1 board with 10x10 SPARTAN FPGA Array [Enterpoint, 2016]

The observational evidence presented to formulate the Concurrent Hypothesis include the facts that the estimated computational load of the **Gas Flow Model** amounts to 0,6 petaflop while Chapter 5 showed the maximum theoretical throughput of the Planar Array Processor to be 5,75 Petaflop based on manufacturers data. To exploit this massive processing power the electrical architecture of the Precision Rocket Launcher and the form of the Taylored Navier Stokes equation have to be considered. Additional observational evidence include a range of FPGA devices that might be suitable for implementing the Planar Array Processor, data types for implementing the **Gas Flow Model**, suitability of the Serial Peripheral Interface (SPI) mechanism as Nearest Neighbour Interface between each adjacent FPGA device pair and the need for determinism and predictability to certify the PRL for flight.

Finally, various methodologies and tools for developing such a large circuit are presented, including the use of Hardware Development Language (HDL), Simulink as Integrated Development Environment (IDE), the Quartus II compiler and the Model Based Design (MBD) paradigm.

No specific example of a circuit similar to the **Concurrent Model** could be found in the open literature other than descriptions of some very large supercomputers such as the IBM Blue Gene/P and Tianhe-2. One example comparable to the problem here is the **Merrick1** board as illustrated in Figure 150. The component layout of the Merrick1 board suggests that somebody in industry had similar ideas to the concepts as developed in this study. Various authors describe the use of FPGA's in computation-intensive applications such as communications and radar, but actual circuits are seldomly shown.

Questions are asked as to what circuit topology would be best suited for implementing the algorithm as described in paragraphs 4.5.4 thru 4.5.8 of Chapter 4, in particular decentralization of functions to circumvent the Processor, Memory and Interface Bottlenecks. In addition, the use of Dual Port Ram to ensure the independent operation of the the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes of the **Gas Flow Model** is also considered. Finally, questions are asked as to what data type would yield the best mechanism for representing the variables of such a system.

Having considered the observational evidence this chapter formulates the Concurrent Hypothesis which states that it should be possible to construct a dedicated circuit that represents millions of cells running concurrently, each containing a copy of the solver as described in paragraphs 4.5.4 thru 4.5.8 and all of them coordinated with high repeatability and predictability using only the resources of the Precision Rocket Launcher as defined in Chapter 5. The Concurrent Hypothesis also states that it should be possible to run this model in real time. Finally, the Concurrent Hypothesis states that it should be possible to construct and test such a circuit using Hardware Development Language (HDL) within the Simulink Integrated Development Environment (IDE).

Some predictions based on the Concurrent Hypothesis are made, the main prediction being that the **Concurrent Model** can be implemented with the Planar Array Processor and resources of the Precision Rocket Launcher, such that this circuit can be commutated, probed and grid hopped at a rate in excess of 10 kHz. This prediction is tested by generating a detailed design of the **Concurrent Model** and embedding it in the Precision Rocket Launcher.

Note: *The **Concurrent Model** as presented in this chapter represents the design of a dedicated circuit that implements the functionality as described in Chapters 0 and 5. A detailed design of this circuit was required in order to prove the Concurrent Hypothesis. This part of the study required extensive effort over a period of 2 years, producing a large quantity of detail which cannot be repeated in whole here. Only the main concepts used and conclusions reached during the design are described here.*

This chapter makes four main contributions to the knowledge database. First, it proves that there is in existence at least one circuit that can predict the downwash flow field of a helicopter at an iteration frequency in excess of the 10kHz iteration frequency as specified in paragraph 5.5.5. Second, it shows that Moores law has progressed to a point where such a circuit can be implemented with a planar array of contemporary FPGA devices packaged within the constraints of the Precision Rocket Launcher, albeit using the best top-end FPGA available from Intel. This would have been impossible a few years ago and the result here serves as confirmation of the trend identified by *Kurzweil* [85] and illustrated in Figure 123. Third, it shows that although the **Concurrent Model** is very fast, it requires a quantity of resources so large that it exceeds the limits of the Precision Rocket Launcher as defined in Chapter 5. This leads to the conclusion that a further trade-off between dedication and re-use is required to implement such a circuit – finding such a trade-off then becomes the purpose of Chapter 7. Fourth, it reveals a serious mismatch between multiplier count and FPGA fabric as found in contemporary FPGA devices (no exception to this rule could be found in the open literature). Finally, it shows that the

limitations of Amdahl can indeed be overcome and that expansion of the number of processors (cores) used in the Array Processor is not limited by the housekeeping function at all (the real limit is researched in Chapter 7).

6.1 Resources and tools (Make observations)

The electrical architecture and throughputs of the Host Processor as coordinater of the Precision Rocket Launcher and the Planar Array Processor as specialized front-end are defined in Chapter 5. A simplistic model which assumes the Planar Array Processor to be implemented with device GX2800 from INTEL indicates that such a solution would have an absolute maximum throughput of 5,75 Petaflop assuming each multiplier to be fully utilized at a clock rate of 800 MHz. The actual throughput achieved when implementing the **Downwash Model** will be well below this due to various limitations such as circuit topology and propagation delays within the FPGA's.

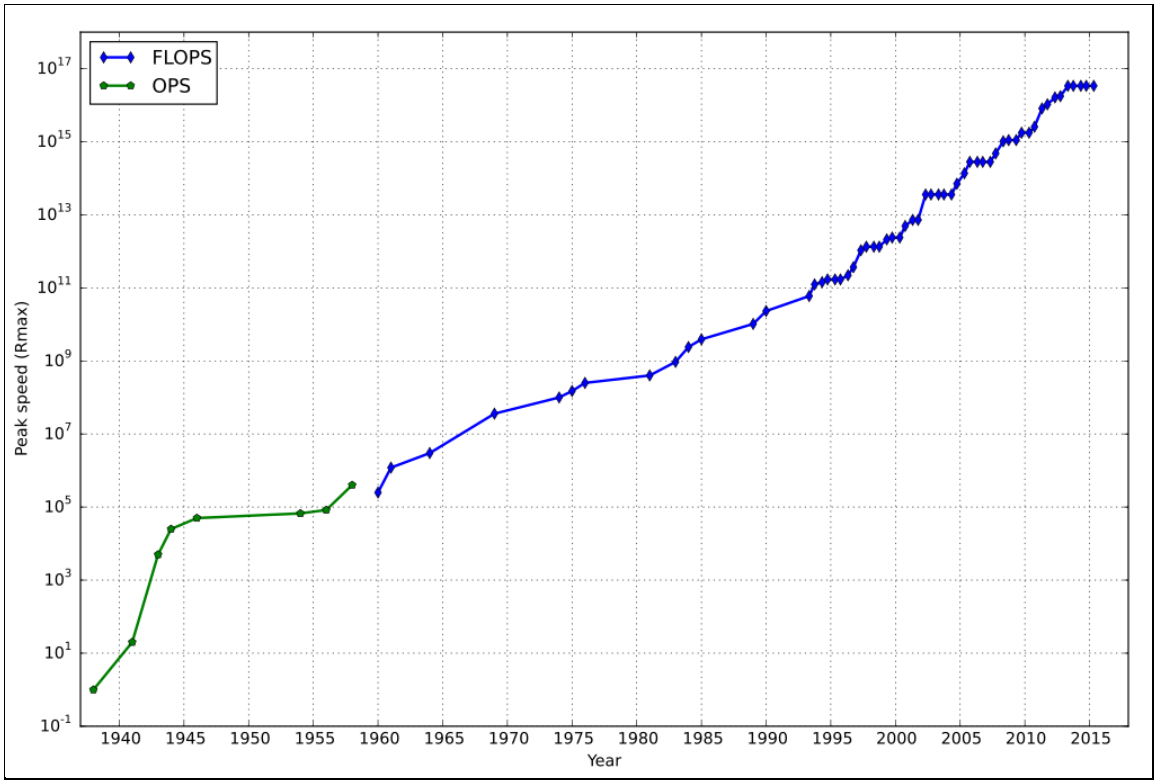


Figure 151. Evolution of supercomputer throughputs [Doege, 2014]

Assuming the Planar Array Processor to have an maximum throughput of 5,75 petaflop it is clear from Figure 151 that the Planar Array Processor as defined by this study is a petascale computer that would have ranked among some of the most powerful computers as at 2010. Note that the achievable throughputs (*Rmax)* as illustrated in Figure 151 was derived with the so-called LINPACK benchmark [113] which calculates the product between a lower triangular matrix and an upper triangular matrix. For comparison purposes, this is similar to calculating the inverse of matrix **A** in Equation 105 through Gaussian elimination.

Inspection of Figure 134 shows that the Planar Array Processor comprises an array of FPGA devices controlled by a processor referred to as the **Array Coordinator**. Each FPGA device contains an internal array of **cores** embedded within the fabric of the device. Each core is a dedicated processor which implements the Gas Flow, Commutation, Probing and Grid Hop processes for a stack of **cells** in accordance with the scheme as described in paragraphs 5.5.1, 5.5.2, 5.5.3 and 5.5.4.

6.1.1 Commutation Bus

Coordination and control of the cores is performed by a central state machine located within the Array Coordinator. This yields a constant and predictable control cycle which requires no housekeeping, thus avoiding the limitations of Amdahl. To this effect, control and data signals are exchanged between Array Coordinator and cells via a **Commutation Bus** which comprises a number of discretes in conjunction with a dedicated address bus and a dedicated data bus. Definition of the Commutation Bus is open and left to the designer who must ensure that data represented in accordance with some or other fixed point or floating point notation can be exchanged via the bus at very high rate. Choice of the notation used is a design decision with considerations as described in 6.1.4.

6.1.2 Nearest Neighbour Interfaces

Located between each FPGA device and its four immediate neighbours are four "Nearest Neighbour" full-duplex serial interfaces with structures as illustrated in Figure 134. These interfaces distribute the states of periphery cells within a given FPGA to corresponding periphery cells within adjacent FPGA's while the states of internal cells are exchanged via the fabric of the FPGA.

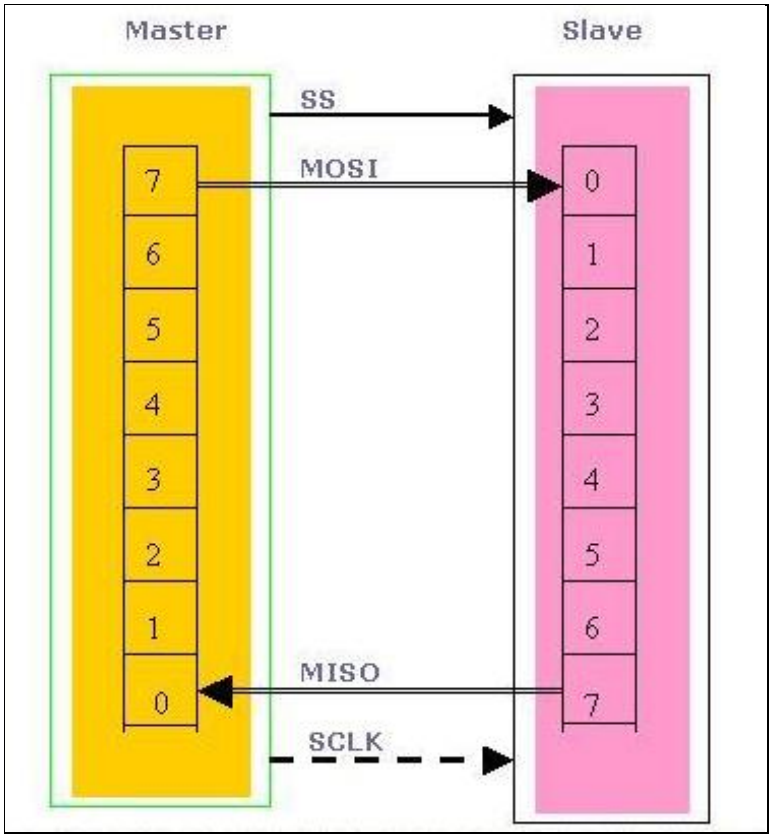


Figure 152. Basic structure of Serial Peripheral Interface [EEHERALD, 2016]

Exchanging this information between FPGA's is necessary for calculating the spatial derivatives of the automata network as defined in paragraph 4.5.4(d). The interfaces are serial to conserve space on the Wrap-Around PCB (refer paragraph 5.5) and is based on the Serial Peripheral Interface (SPI) which forms a circular shift register with main components as illustrated in Figure 152. SPI is well suited for this purpose because it is not constrained by a formal specification [114]. Hence, using the fabric of two adjacent FPGA's the shift registers of Figure 152 can be made any length as may be required by the design. In addition, the single-ended *MOSI* and *MISO* signals can be replaced with Low Voltage Differential Tranceivers, allowing transfer rates up to 14,1 Gbits/sec (see Table 7). Note that in the Planar Array Processor as defined in this

study all the Nearest Neighbour interfaces share a common *SCLK* clock generated by a central state machine, ensuring lockstep synchronization of all the cells of the array. Note that the *SS* signal is absent since each Nearest Neighbour Interface is dedicated.

6.1.3 Choice of device type

For the purposes of this study a range of Intel FPGA devices as listed in Table 7 were evaluated as possible candidates for implementing the Planar Array Processor. Devices chosen cover the midrange to top-end part of the spectrum of devices that were thought to have a chance of being suitable for the task. The consideration here was that numerous midrange (small and cheap) devices might provide the same computation power as a few (large and very expensive) top-end devices. This approach was necessary because at the onset of research as described in this chapter the class of FPGA required to implement the **Downwash Model** and run it in real time was completely unknown.

Table 7. FPGA devices considered for implementing Array Processor [A Landman, 2017]

Device Family	Main features
Cyclone V	▪ 113560 Adaptive Logic Modules (ALM) implemented with 28 nm technology ▪ Maximum fabric clock frequency of 550 MHz ▪ Device 5CEFA9F23C7 contains 684 hardware multipliers, 342 DSP blocks and 12,49 Mbit RAM ▪ Device 5CEFA9F23C7 contains 12 Low Voltage Differential Tranceivers that can be clocked at 3 Gbits/sec. This may be suitable for implementing high-speed Serial Peripheral Interfaces (SPI) between adjacent FPGA devices ▪ Device 5CEFA9F23C7 available as 19x19 Super Fine Ball Grid Array (SFBGA) with 484 balls at 0,8 mm pitch. Hence, a 10x10 array of 5CEFA9F23C7 devices will fit onto APM assuming 2 mm inter-device gap
Arria 10	▪ 250540 Adaptive Logic Modules (ALM) implemented with 20 nm technology ▪ Maximum fabric clock frequency of 500 MHz ▪ Device 10AS066H2F34E1SG contains 3376 hardware multipliers, 1687 DSP blocks and 43,64288 Mbit RAM ▪ Device 10AS066H2F34E1SG contains 48 Low Voltage Differential Tranceivers that can be clocked at 17,4 Gbits/sec. This is suitable for implementing high-speed Serial Peripheral Interfaces (SPI) between adjacent FPGA devices ▪ Device 10AS066H2F34E1SG available as 35x35 mm Super Fine Ball Grid Array (SFBGA) with 1152 balls at 0,8 mm pitch. Hence, a 5x5 array of 10AS066H2F34E1SG devices will fit onto APM assuming 8,25 mm inter-device gap
Stratix V	▪ 262400 Adaptive Logic Modules (ALM) implemented with 28 nm technology ▪ Maximum fabric clock frequency of 800 MHz ▪ Device 5SGSMD8K1F40C2 contains 3926 hardware multipliers, 1963

	DSP blocks and 52,57216 Mbit RAM ▪ Device 5SGSMD8K1F40C2 contains 48 Low Voltage Differential Tranceivers that can be clocked up to 14,1 Gbits/sec. This is suitable for implementing high-speed Serial Peripheral Interfaces (SPI) between adjacent FPGA devices ▪ Device 5SGSMD8K1F40C2 available as 40x40 mm Super Fine Ball Grid Array (SFBGA) with 1517 balls at 1,0 mm pitch. Hence, a 5x5 array of 5SGSMD8K1F40C2 devices will fit onto APM assuming 2,0 mm inter-device gap
Stratix 10	▪ 933120 Adaptive Logic Modules (ALM) implemented with 14 nm technology ▪ Maximum fabric clock frequency of 800 MHz ▪ Device GX2800 contains 11520 hardware multipliers, 5760 DSP blocks and 229 Mbit RAM ▪ Device GX2800 contains 96 Low Voltage Differential Tranceivers that can be clocked up to 28,3 Gbits/sec. This is suitable for implementing high-speed Serial Peripheral Interfaces (SPI) between adjacent FPGA devices ▪ Device GX2800 available as 42,5x52,5 mm Super Fine Ball Grid Array (SFBGA) with 1152 balls at 1,0 mm pitch. Hence, a 5x5 array of GX2800 devices will fit onto a slightly larger APM assuming 2 mm inter-device gap

Note that other devices suitable for implementing the Array Processor are manufactured by other manufacturers. Since they are similar in function and shape, selecting these devices would also have led to similar results and conclusions. The main driver for choosing the Intel range of devices was the availability of a suitable Integrated Development Environment (IDE) and Hardware Development Language (HDL) compiler that could interface directly with Simulink.

6.1.4 Fixed point versus Floating Point notation

One of the major decisions to be made in designing an embedded system is the choice of data format for representing variables and performing mathematical operations of the system. *Smith* [115] shows that in the case of Digital Signal Processors this choice revolves mainly about a trade-off between cost and accuracy. Until recently the rule of thumb was that fixed point notation would be the preferred notation for devices produced in large numbers (such as cellphones) where production cost is the main driver. In contrast, floating point notation would be the preferred choice for expensive devices (such as a navigation computer) where accuracy is the main driver. For cellphones the small cost advantage of fixed point notation can mean the difference between success and failure. For navigation computers accuracy can mean the difference between an aircraft reaching its destination with enough fuel to land or ditching out at sea.

It is evident from the analysis by *Smith* [115] that the rule of thumb would indicate floating point notation to be the preferred choice for implementing the Precision Rocket Laucher and Array Processor. This would have been true if the Precision Rocket Laucher contained only a few DSP's and FPGA's. In reality the Array Processor will contain several hundred top-end FPGA's that have very high unit costs (each about the cost of a small car). In such a case the cost of FPGA devices used to implement the Array Processor constitutes a major fraction of the overall cost of the Precision Rocket Launcher. Hence, using 16 bit fixed point notation versus 64 bit floating point notation can mean a production cost difference of many millions of Rand for the PRLS system.

Deciding on a suitable notation for the Precision Rocket Launcher therefore implies finding a fixed point word width where the PRLS would achieve the ballistic accuracy as hypothesized in Chapter 1. The performance of such a system should be compared with the same system implemented in single and double precision floating point notations and a suitable choice made (this is one of the main contributions of this chapter).

Choosing between fixed point and floating point for implementing an embedded application is no simple task. *Corless* and *Ananthan* [116] describe how this can be done for a digital filter in an automotive application within the Simulink environment following the Model Based Design (MBD) paradigm. Their approach is to start the design process by building a model of the required system using double-precision floating point notation. This yields a model with the best resolution and dynamic range (i.e. no overflows) which then serves as reference for comparison with the same model implemented in fixed point notations at different word lengths. After selection of a suitable word length the verified fixed point model is then converted into C language for compilation and installation on the intended hardware. For the purposes of this study the same process is followed, except that the model is converted into Hardware Development Language (HDL) for compilation and installation on an FPGA device instead of a DSP device.

6.1.5 Central coordination and determinism

The primary purpose of a Combat Helicopter is Close Air Support (CAS) as described in paragraph 1.1.3. This means that on occasion FZ90 Rockets will be delivered as close as 20 meter from own troops which in turn means that the PRLS is a safety critical system. Hence, the software of the Precision Rocket Launcher would be categorized as Class A in accordance with *DO-178C* [20]. Note that the Array Processor does not contain software *per se*, but that ultimately the **Downwash Model** will be implemented in a circuit comprising quadrillions of physical transistors. Nevertheless, the circuit will be developed using Hardware Description Language (HDL) in a process which has characteristics similar to normal software development using a High Level Language such as C.

To certify such a system requires many things, one of which is to prove that all modes and combinations of the software (or circuit in the PRLS case) are both necessary and without error. This can be a mammoth task which calls for simplification, thus triggering the need for determinism and predictability of the Downwash Algorithm as embedded in the Array Processor.

To satisfy this need and the need for speed the design approach as followed in this thesis was establishment of a circuit in which all the cores run synchronous and in parallel in accordance with a fixed state transition sequence as directed by a state controller located in the Main Electronics Unit (see paragraph 5.5.8).

6.1.6 Development tools

For the purposes of this study the Integrated Development Environment (IDE) used for developing a circuit to represent the **Concurrent Model** was Simulink in conjunction with the Fixed Point Tool (which converts a floating point model into a fixed point model) and the HDL Coder (which generates HDL files directly from the Simulink model). Note that the HDL Coder in turn calls the Quartus II compiler from Intel (previously Altera), creates a Quartus II project with the HDL files installed and finally triggers compilation of the HDL files into a binary file for downloading into the FPGA.

6.2 Quadrillions of transistors (Think of interesting questions)

Ultimately the strategy of the PRLS for calculating the downwash of the helicopter in real time is to embed the **Downwash Model** in a huge circuit comprising about $1{,}875.10^{12}$ transistors that will all operate concurrently. Is it even possible for such a circuit to be designed and developed with contemporary electronics and expect it to operate in a harsh environment?

Answering the question above is of course one of the objectives of this study, and there are good reasons to believe that it may well be possible. One reason is that miniaturization of electronics over the past century has progressed to the point where a contemporary FPGA now contains 30 billion transistors (refer paragraph 5.1.3) and the re-use operating point has dropped to very low values. Another reason is that production and packaging techniques of electronics has progressed to the point where it is now possible to interconnect, pack and cool many of these devices in strange shapes such as the periphery of the Precision Rocket Launcher (refer paragraphs 5.5 and 5.5.9). Finally, the behaviour of the circuit can be simulated with great precision beforehand so that the circuit can be designed, optimized and errors eliminated even before the circuit is constructed.

6.3 The Concurrent Hypothesis (Formulate hypothesis)

Considering all the observational evidence as presented above we postulate that it should be possible to design a bespoke circuit compatible with the re-use operating point as implied by the Planar Array Processor and resources of the Precision Rocket Launcher as defined in Chapter 5. It should be possible to design this circuit such that it comprises hundreds of thousands of cells running concurrently with each cell containing a copy of the solver as described in paragraphs 4.5.4 thru 4.5.8. It should be possible to coordinate all of these cells with a central state machine that performs a known and repeating state transition sequence that avoids the housekeeping limit of Amdahl. It should also be possible to implement this circuit in fixed point notation and run it at an iteration rate in excess of 10 kHz. Finally, it should be possible to design and test this circuit using Hardware Development Language within the Simulink Integrated Development Environment.

6.4 Concurrent Predictions (Develop testable predictions)

If the Concurrent Hypothesis is true the **Concurrent Model** will be capable of estimating the downwash flow field of a helicopter within a 100x100x100 meter flight box at a temporal resolution smaller than 0,1 millisecond and a spatial resolution smaller than 200 mm when implemented on the Precision Rocket Launcher.

6.5 Implementing the Concurrent Model (Gather data to test predictions)

The design process was initiated by assuming that all the FPGA devices as listed in Table 7 are big enough to contain at least two cores (the amount of FPGA resources required to implement a single core was still unknown at this stage). Under this assumption, various attempts were made to express the algorithms of Chapters 4 and 5 in terms of a Data Flow Block Diagram suitable for implementation on the Precision Rocket Launcher using FPGA's containing an even number of cores.

A typical example (simplified) of such a Data Flow Block Diagram is illustrated in Figure 153, showing how a 4x6 Planar Array Processor could be implemented within the constraints of the Precision Rocket Launcher. In this example use is made of two FPGA's each of which contains four cores with internal organization as described below.

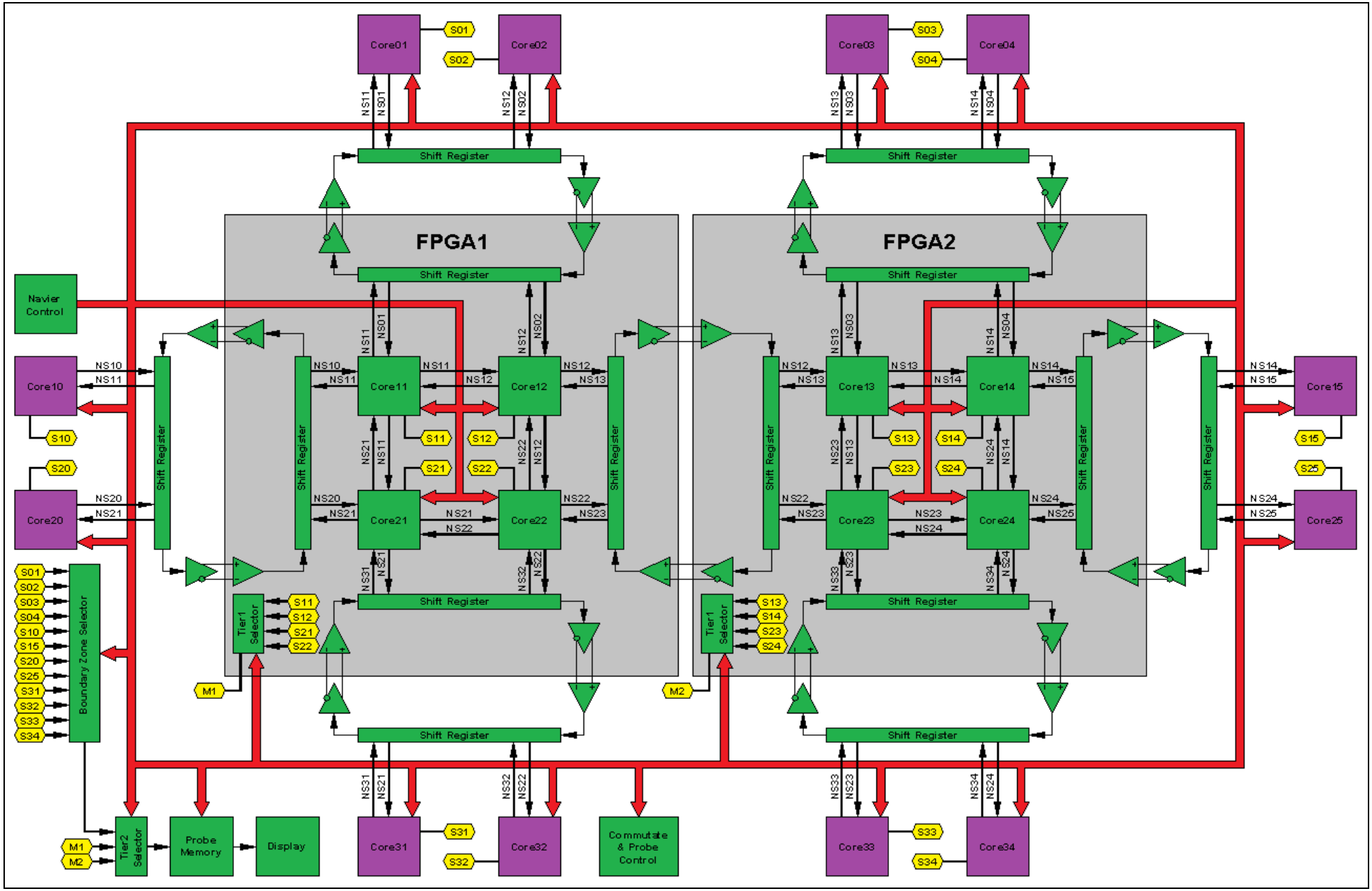


Figure 153. Overall DFBD for 4x6 Concurrent Model example [A Landman, 2018]

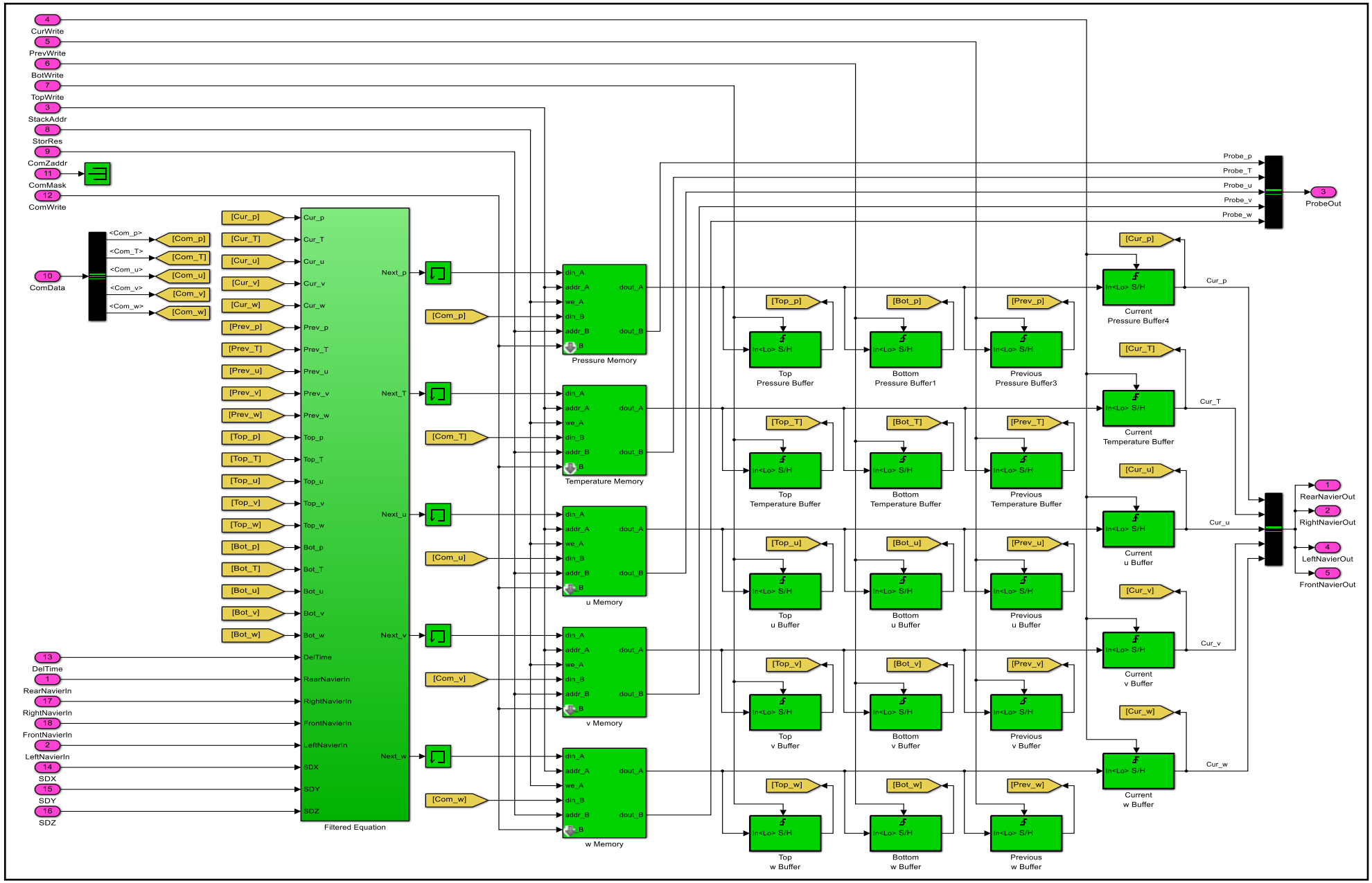


Figure 154. Internal organization of a Concurrent Model core [A Landman, 2016]

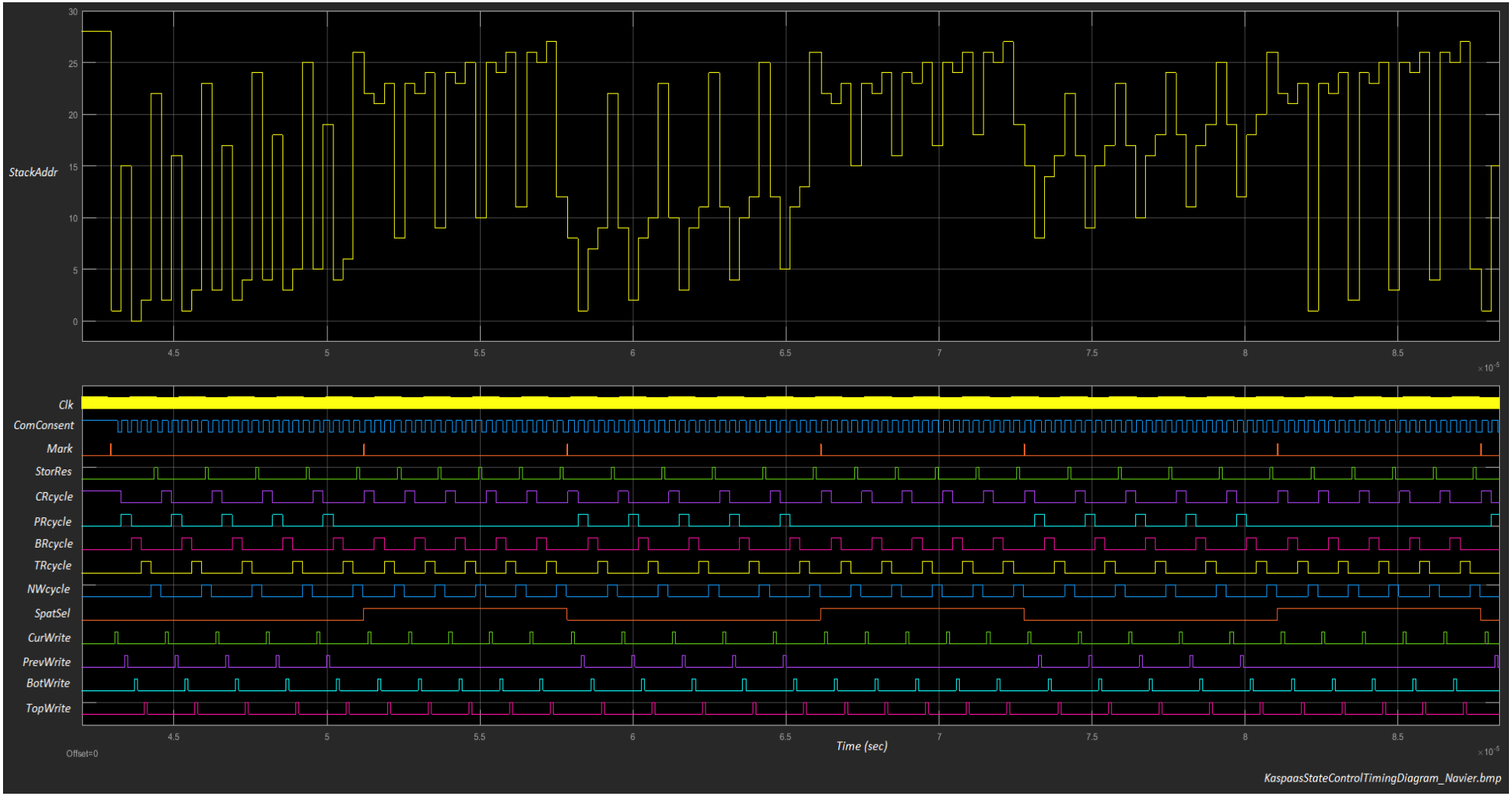


Figure 155. Example of signals from Navier Control block of Concurrent Model with 7x7x7 Flight Box [A Landman, 2016]

The purple blocks Figure 153 represent boundary cores, each of which is essentially memory that can be overwritten with the Commutation Process but otherwise looks like a normal core. Note the use of shift registers to implement nearest neighbor interfaces with the Serial Peripheral Interface configuration. This is necessary because the number of bits involved exceed by far the number of I/O pins available on the FPGA devices as listed in Table 7.

The internal organization of a single core is illustrated in Figure 154. Note that the 525x525 array of cores need not be designed in its entirety due to the repeating nature of the circuit – designing a DFBD of a single core will suffice. Under these conditions the objective was to design a DFBD which:

a) Calculates the local spatial derivatives in accordance with Equation 100, Equation 101, Equation 102 and Equation 103.

b) Uses the spatial derivatives of (a) to calculate the temporal derivatives in accordance with Equation 105.

c) Integrates the temporal derivatives of (b) in accordance with Equation 109.

d) Filters the result of (c) in the time domain using Equation 110.

e) Filters the result of (d) in the spatial domain using Equation 111.

f) Contains Random Access Memory (RAM) to store the result for use in the next computation cycle, such that a circular buffer similar to that of the Matlab program of paragraph 4.5.8 is established.

g) Allows the contents of RAM to be overridden (commutated) or extracted (probed) without disturbing the main computation cycle.

In general, the inputs at top left of Figure 154 come from the Commutation Bus, inputs from bottom left come from neighbouring cells (via FPGA fabric or Nearest Neighbour Interfaces), outputs at top right feed into the Commutation Bus and outputs at bottom right go to neighbouring cells (via FPGA fabric or Nearest Neighbour Interfaces).

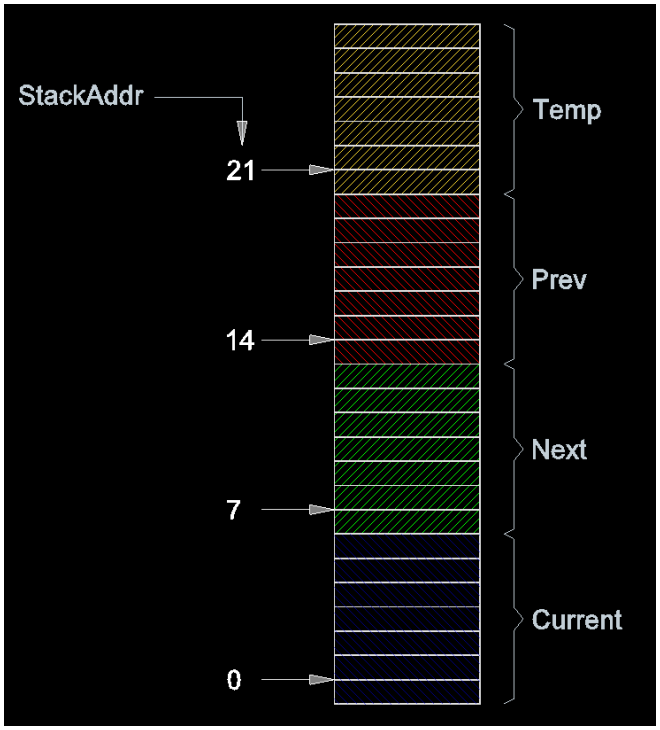


Figure 156. Memory map of Stack for 7x7x7 Flight Box example [A Landman, 2016]

Note: *The delay blocks in Figure 154 are necessary to break the algebraic loops formed by the circuit – they are required for simulation purposes but have no effect on the result.*

Five dual-port RAM blocks (embedded within in the FPGA fabric) are used to implement a **Stack** containing the states of cells as illustrated in Figure 136 and memory map as illustrated in Figure 156. Separate RAM blocks are used so that pressure *p*, temperature *T* and the three velocity components *u*, *v* and *w* of the current cell can be written or read simultaneously. By connecting the *addr_A* inputs of all the RAM blocks to the *Stack_Addr* signal the contents of the same location within all the RAM blocks can be transferred to a set of buffers at the same time. This read and store process is repeated until all the variables associated with the current cell are extracted from RAM and loaded into 20 separate buffers as illustrated in Figure 154. The outputs of these buffers are then simultaneously fed into the **Filtered Equation** block to solve the Taylored Navier Stokes equation as described in Chapter 4, yielding the next estimate of state variables of the current cell which are then copied back into RAM to serve as current estimate during the next iteration.

For the purpose of this study this basic sequence of events as described above to calculate the next estimate of state variables for a given cell is referred to as a **Minor Cycle**. A collection of Minor Cycles running in sequence to estimate of state variables for all the cells in the stack is referred to as a **Major Cycle**.

In addition to the basic need for RAM as described above there is also the need to perform the second-order temporal and spatial filters as described in paragraphs 4.5.7 and 4.5.8. To implement the temporal filter each cell requires three data sets reflecting the previous estimate as calculated two iterations ago, the current estimate as calculated one iteration ago and the next estimate as currently being calculated (see paragraph 4.5.6). These data sets are stored in RAM with memory map reflecting four partitions as illustrated in Figure 156 for a 7x7x7 Flight Box as example. Pointers are used to access and overwrite the contents of RAM in such a way that a circular buffer is formed. Within this arrangement the locations of previous, current and next iterations are determined by the pointers so that data need not be shifted between different locations within RAM.

To implement the spatial filter each Major Cycle is divided into two stages, the first of which calculates a temporally-filtered solution some time Δt into the future and the second of which applies a spatial filter to the temporally-filtered solution. The temporally-filtered solution is stored in the *Temp* partition as illustrated in Figure 156 for further processing during the second stage when the spatially filtered result is written back into the next circular buffer partition. A typical sequence of signals as generated by the **Navier Control** block for controlling a 7x7x7 Planar Array Processor is illustrated in Figure 155. Inspection of this figure illustrates the fact that each Major Cycle comprises two stages and that three variations of Major Cycles emerge as the system cycles through the circular buffer.

6.5.1 Inside the Filtered Equation block

A simplified DFBD of the **Filtered Equation** block is illustrated in Figure 157. As shown, it comprises a number of sub-blocks that implement the process as described in paragraph 4.5. The DFBD's of the sub-blocks are illustrated in Figure 158 to Figure 164. These sub-blocks reflect Equation 101, Equation 102 and Equation 103 for calculating the spatial derivatives of the current cell, Equation 105 for calculating the partial derivatives with respect to time of the current cell, Equation 109 for integrating these derivatives and Equation 110 and Equation 111 for filtering the result in the temporal and spatial domains.

Note: *Henceforth the first matrix on the right of the equal sign in Equation 105 will be referred to as matrix* ***A*** *and its inverse as matrix* ***B****. The second matrix on right of equal sign in Equation 105 will be referred to as matrix* ***C****.*

Inspection of Figure 157 reveals the brute-force nature of the **Filtered Equation** block. Data is simply fed in on the left and progresses through the circuit until a result appears on the right – the circuit is characterized by 0% re-use and 100% dedication. As a result, the maximum speed of operation is determined by the longest propagation delay through the circuit.

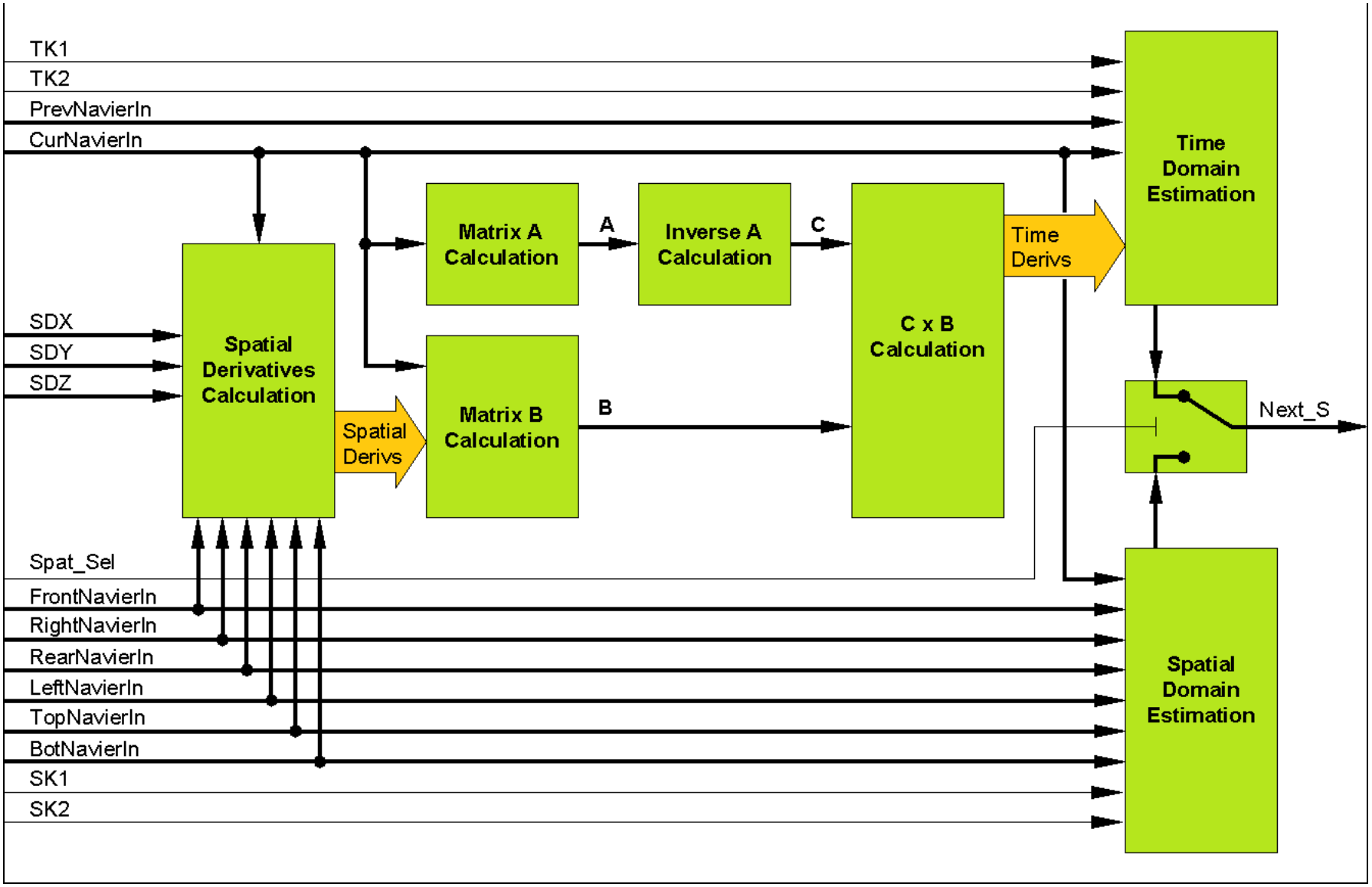


Figure 157. DFBD for the Filtered Equation block of the Concurrent Model [A Landman, 2018]

6.5.1.1 The Spatial Derivatives Calculation block

This block calculates the spatial derivatives as required by Equation 105 for the current cell (as indicated by signal *Stack_Addr*). The DFBD of the **Spatial Derivatives Calculation** block is illustrated in Figure 158. The calculation is performed in accordance with Equation 101, Equation 102 and Equation 103. Combining these equations yields a circuit that can be greatly simplified by eliminating those nodes of the circuit that generate zero outputs as a result of the zero terms in Equation 101.

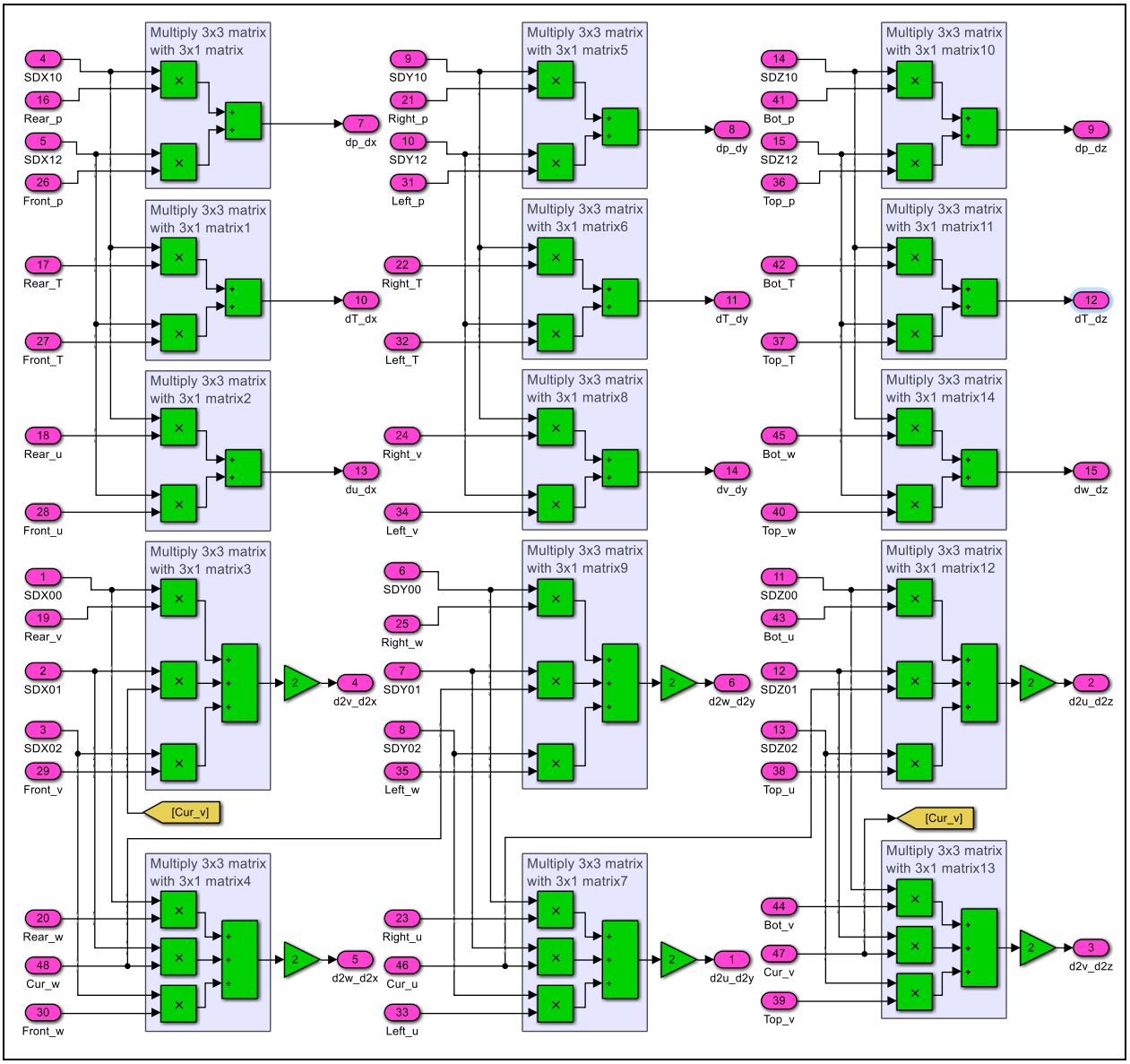


Figure 158. Simplified DFBD for Spatial Derivatives Calculation block [A Landman, 2016

6.5.1.2 The Matrix A Calculation block

This block calculates the value of matrix **A** in Equation 105 for the current cell (as indicated by signal *Stack_Addr*). The DFBD of the **Matrix A Calculation** block is illustrated in Figure 159.

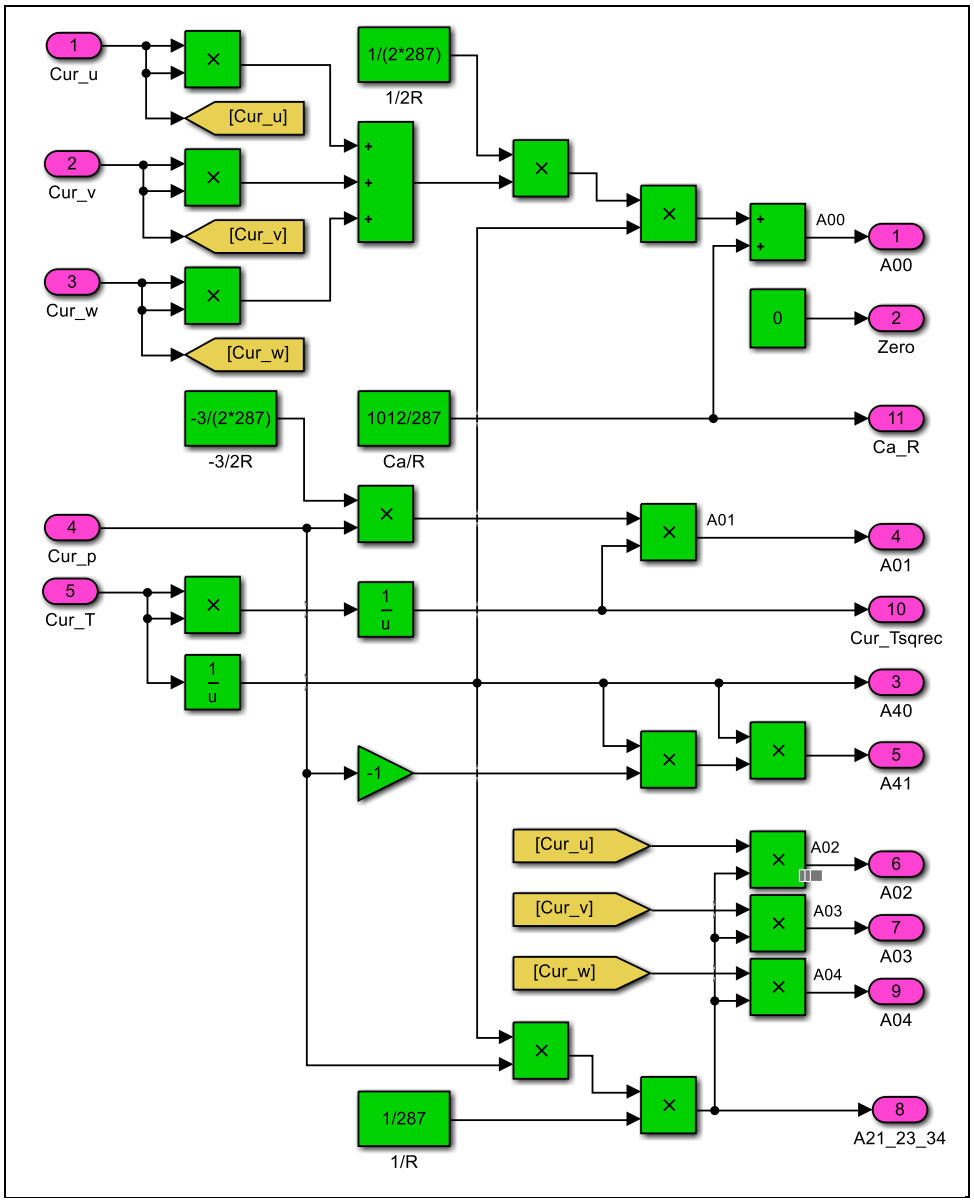


Figure 159. DFBD for Matrix A Calculation block [A Landman, 2016]

6.5.1.3 The Matrix B Calculation block

This block calculates the value of matrix **B** in Equation 105 for the current cell (as indicated by signal *Stack_Addr*). The DFBD of the **Matrix B Calculation** block is illustrated in Figure 160.

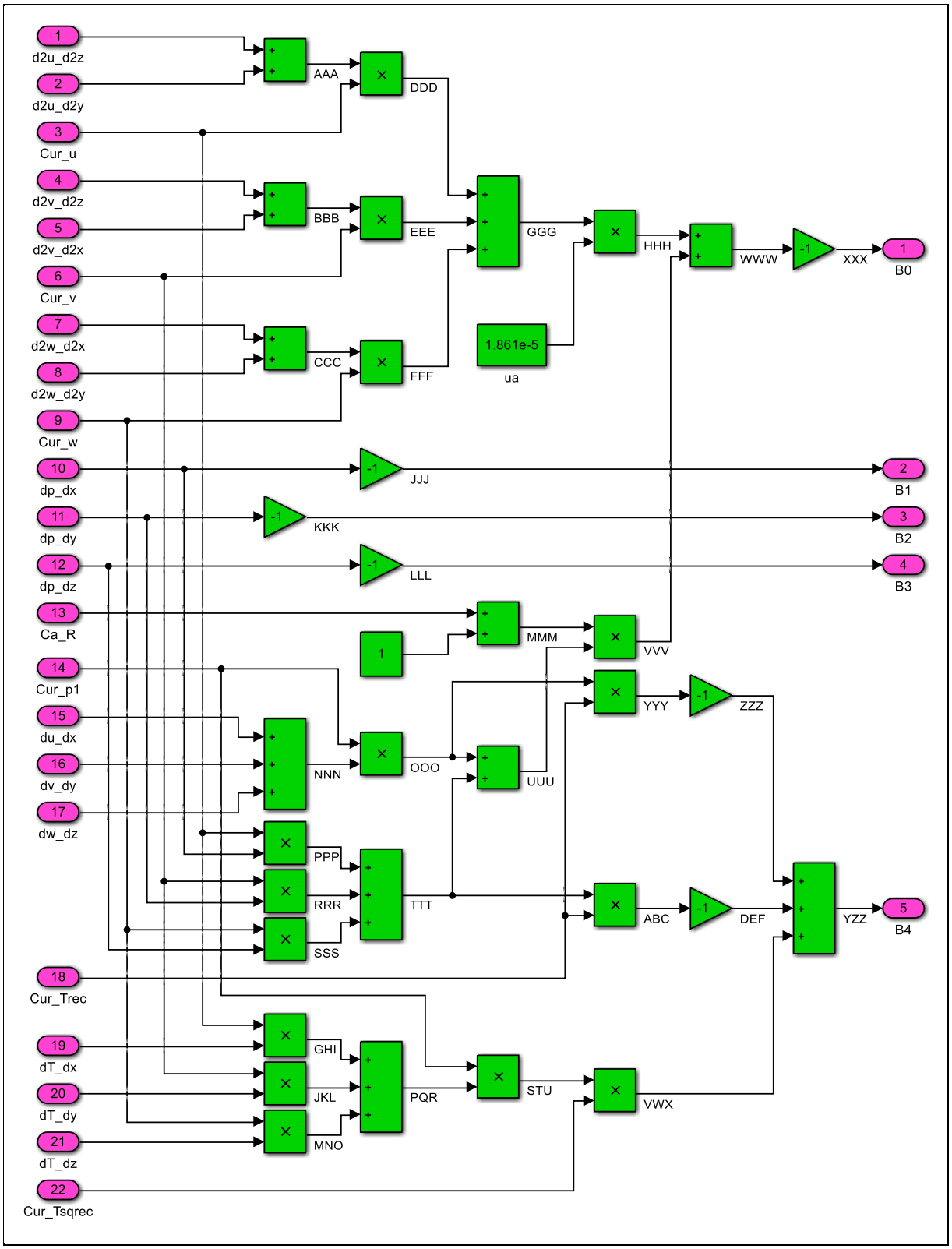


Figure 160. DFBD for Matrix B Calculation block [A Landman, 2016]

6.5.1.4 The Inverse A Calculation block

This block calculates the inverse of matrix **A** in Equation 105 for the current cell (as indicated by signal *Stack_Addr*). The DFBD of the **Inverse A Calculation** block is illustrated in Figure 161. This circuit was greatly simplified by eliminating those nodes of the circuit that remain at zero as a result of the zero terms in matrix **A** of Equation 105.

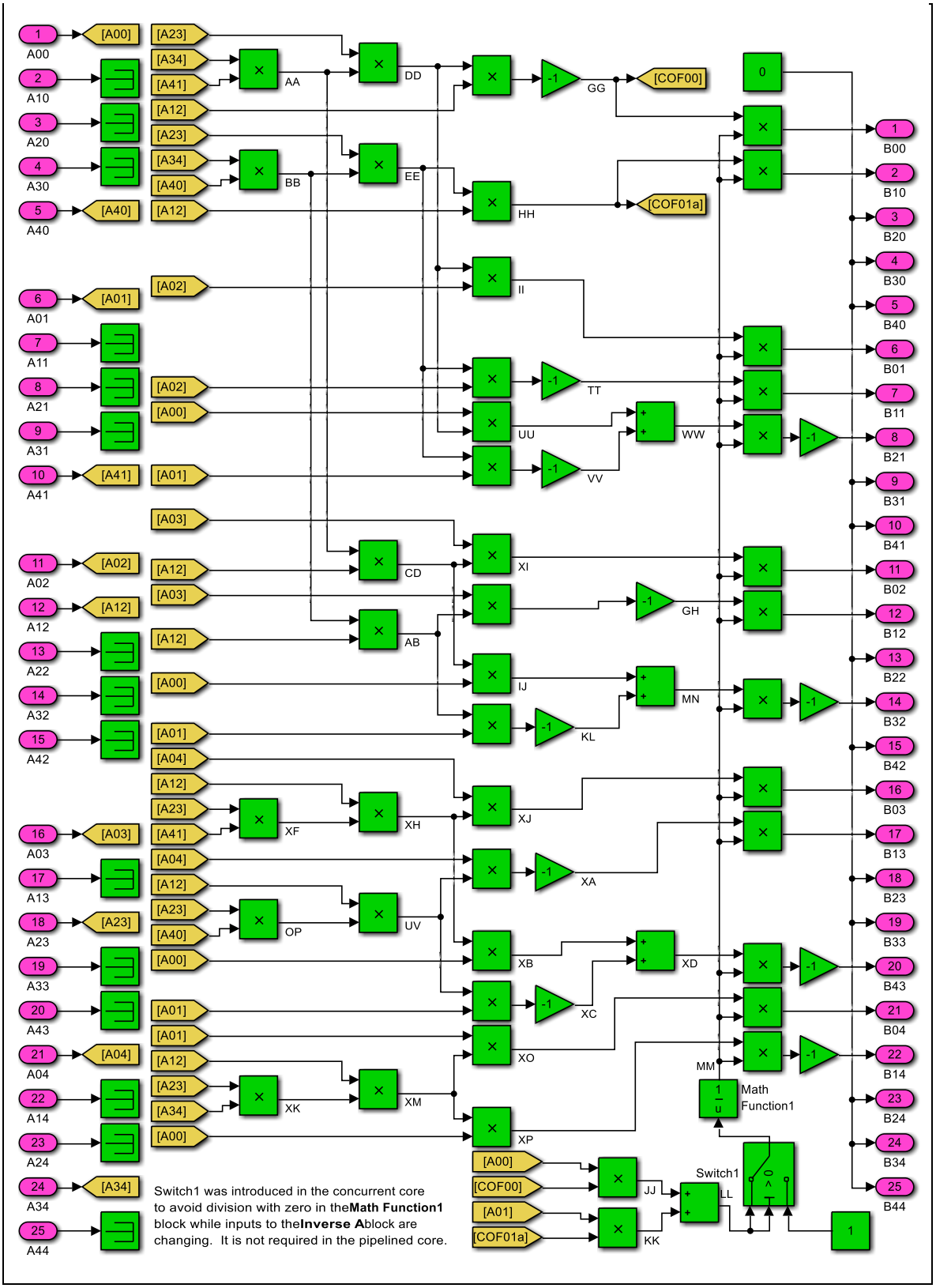


Figure 161. Simplified DFBD for Inverse A Calculation block [A Landman, 2016]

The **Inverse A Calculation** block calculates the inverse of matrix **A** to yield matrix **B** in Equation 105. This block uses the method of co-factors to guarantee a predictable response with a fixed computation time. The method of Gaussian Elimination as described by *Fuller* [117] and *Higham* [118] was specifically avoided here because the detail of row operations required to reduce the matrix into row echelon form cannot be specified in fixed form beforehand. Although such an algorithm can be implemented in dedicated logic with the help of a state machine, the resulting DFBD has variable computation time as shown by *Fang & Havas* [119] which is not desirable for certification of level A software in accordance with *DO-178C* [20].

Some of the inverses as calculated by the **Inverse A Calculation** block while solving the Big Box experiment with the **Concurrent Model** using a 7x7 array were compared with the same inverses as calculated through the method of Gaussian Elimination. The results were the same to at least the fifth decimal which is good enough for the purposes of this study.

It should also be noted here that the solution of Equation 105 is essentially chaotic or on the verge of chaos as found by *Lorenz* [25] as well as *Swinney & Gollub* [27] for similar situations. In such a system small variations in state can be greatly amplified as found by these researchers. In the case of the **Concurrent Model** this can happen if matrix **A** of Equation 105 is close to singular. In various simulations using the **Concurrent Model** the **Inverse A Calculation** block has thus far not showed any sign of causing this type of numerical instability, provided the provisions as described in paragraphs 4.5.6, 4.5.7 and 4.5.8 are complied with.

6.5.1.5 The CxB Calculation block

This block calculates the product between Matric **C** and Matrix **B** in Equation 105 for the current cell (as indicated by signal *Stack_Addr*). The DFBD of the **CxB Calculation** block is illustrated in Figure 162. This circuit was greatly simplified by eliminating those nodes of the circuit that remain at zero as a result of the zero terms in matrix **A** of Equation 105.

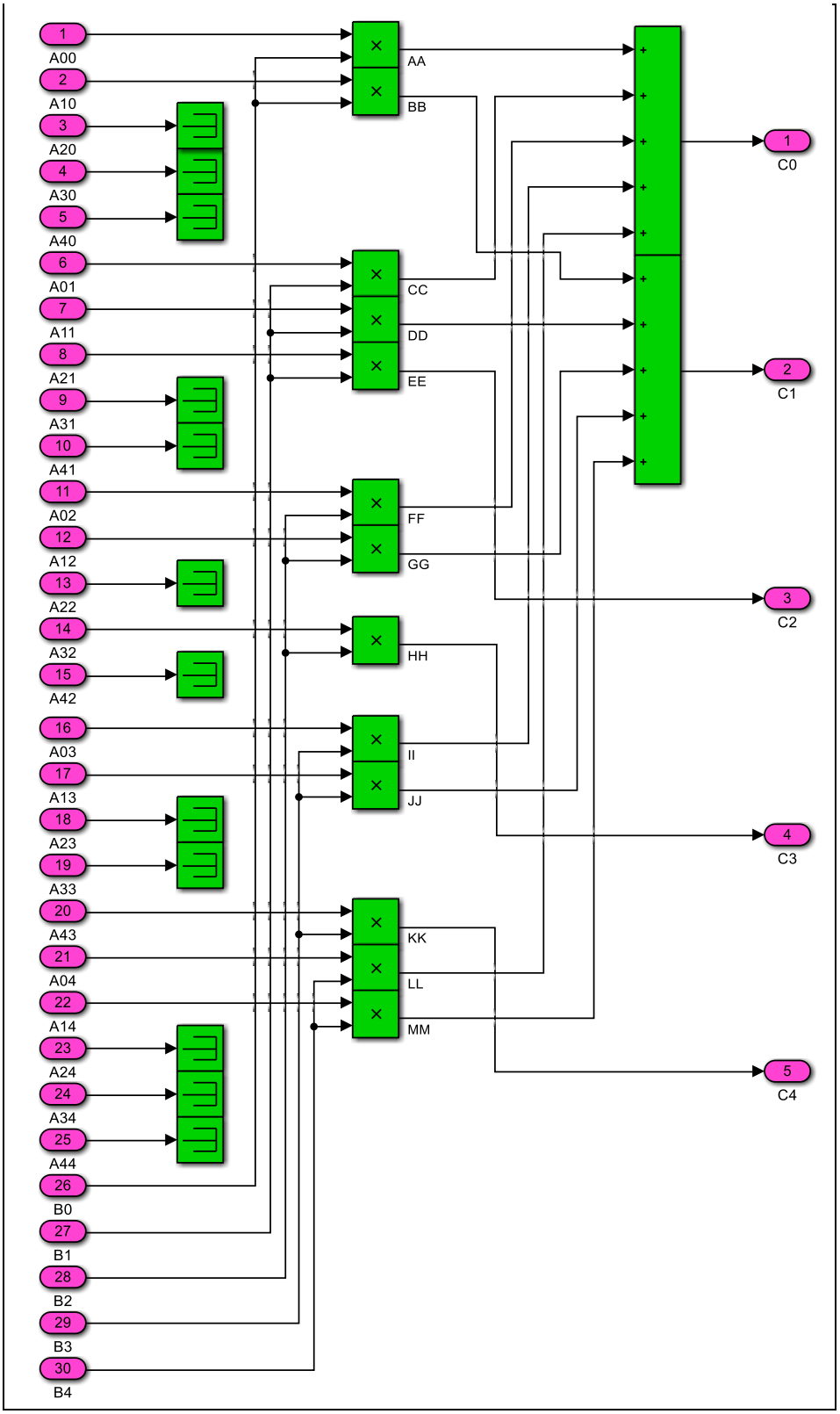


Figure 162. Simplified DFBD for Inverse A Calculation block [A Landman, 2016]

6.5.1.6 The Time Domain Estimation block

This block calculates the initial estimate of state variables in the time domain for the current cell (as indicated by signal *Stack_Addr*) in accordance with Equation 109 and Equation 110. The DFBD of the **Time Domain Estimation** block is illustrated in Figure 163.

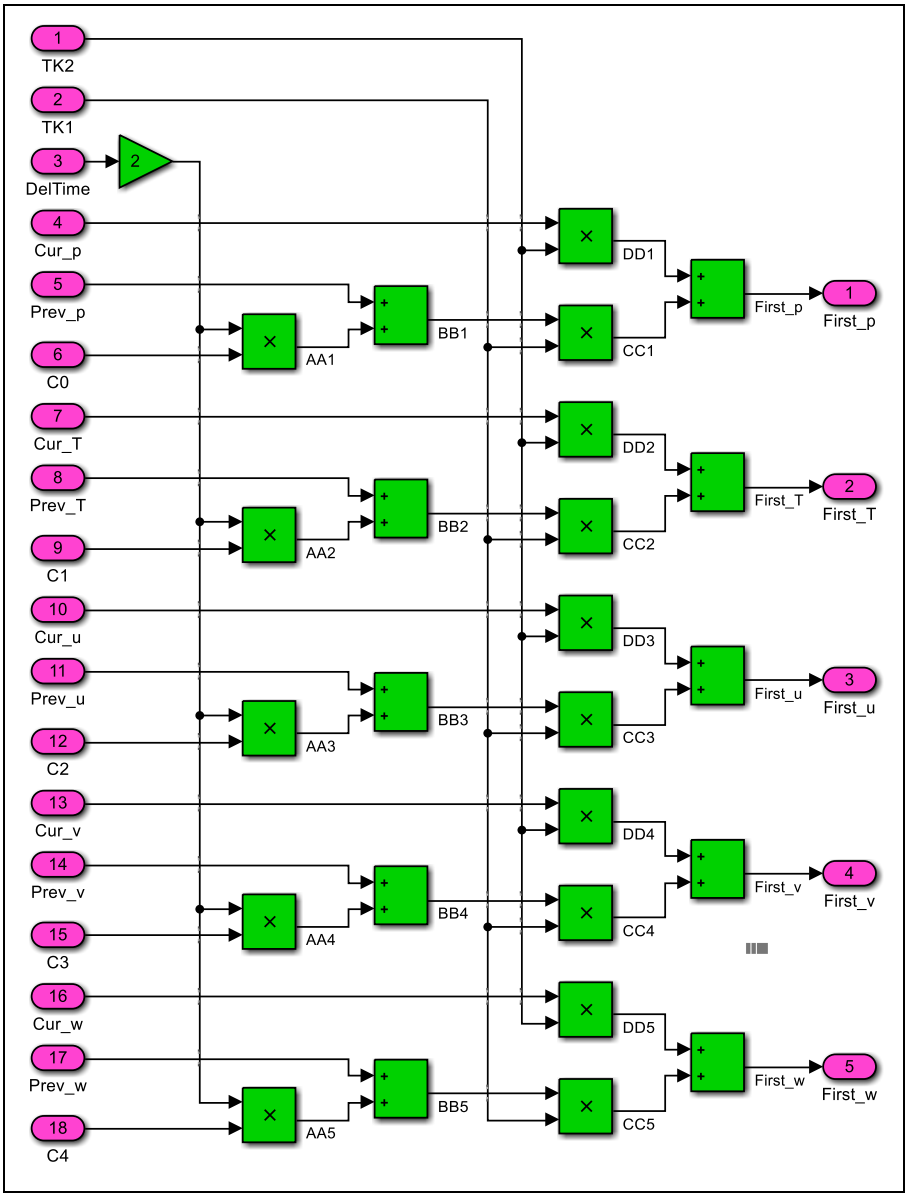


Figure 163. DFBD for Time Domain Estimation block [A Landman, 2016]

6.5.1.7 The Spatial Domain Estimation block

This block calculates the final estimate of state variables in the spatial domain for the current cell (as indicated by signal *Stack_Addr*) in accordance with Equation 111. The DFBD of the **Spatial Domain Estimation** block is illustrated in Figure 163.

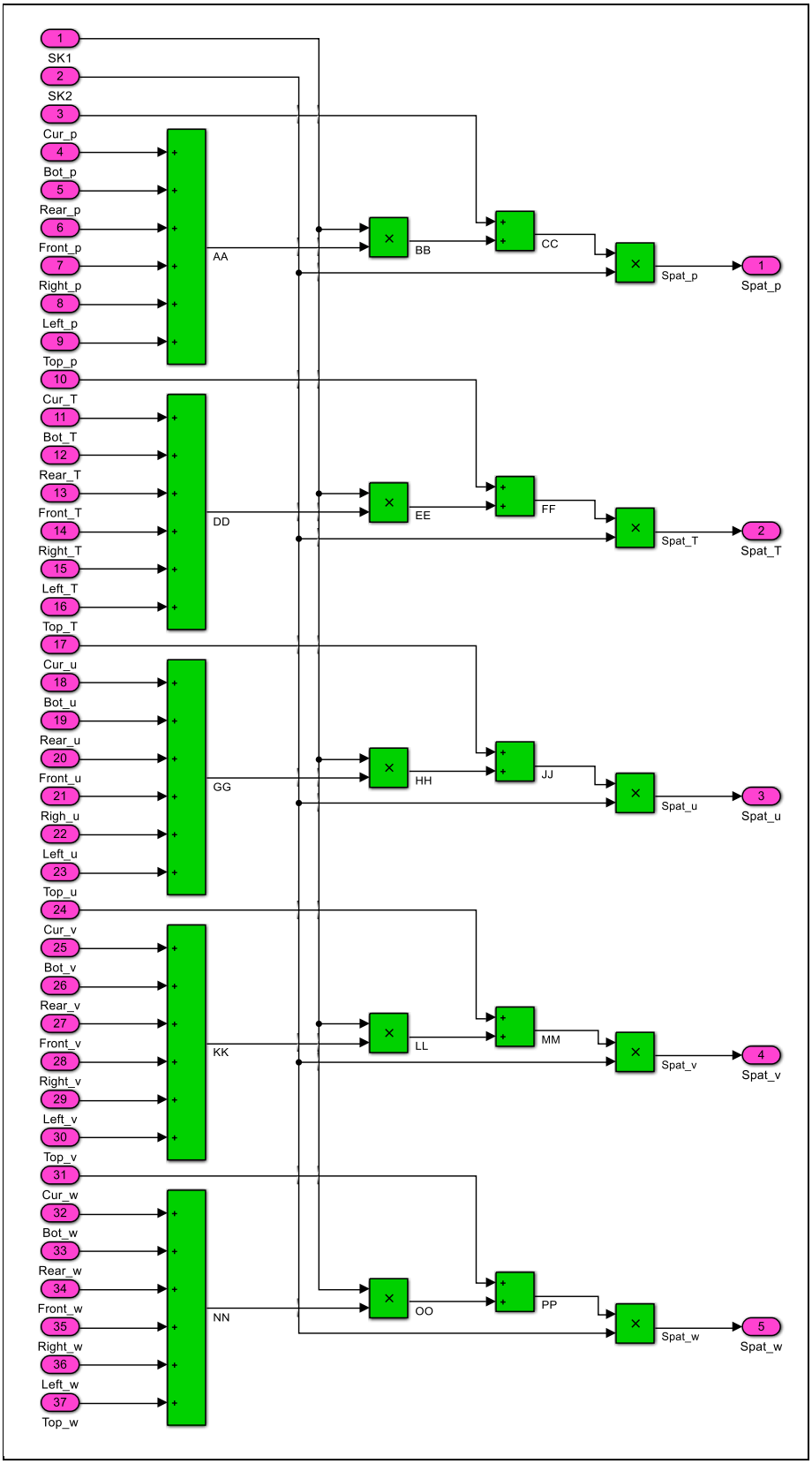


Figure 164. DFBD for Spatial Domain Estimation block [A Landman, 2016]

6.5.2 Verification of the Concurrent Model

Correct operation of the **Concurrent Model** was verified by initializing the model with the same initial conditions as used during the Big Box experiment of paragraph 4.5.8. This made it possible to compare the video sequence (movie) as generated with the Big Box experiment under paragraph 4.5.8 with the video sequence (movie) as generated by the **Concurrent Model**. Both models produced the same result.

The biggest problem of this test was simulation time. Whereas the Matlab script of the Big Box experiment required only a few minutes to model about 500 msec of real world operation, the **Concurrent Model** required several days to do the same. This is because the HDL Coder of Simulink attempts to model the behavior of each basic building block (e.g. RAM, adders and multipliers) in the model in terms of physical considerations necessary to implement the block in FPGA fabric.

Note the objective here was not to produce an exact result, but to confirm that the **Concurrent Model** executes the same mathematics as the Matlab script developed under paragraph 4.5.5. This is all that was required at this stage, since the objective here was to determine if an assembly of Equations 101, 102, 103, 105, 109, 110 and 111 can be clocked at an overall iteration rate of 10kHz when embedded in the fabric of an array of FPGA's.

6.5.3 Converting the Concurrent Model into Fixed Point

Thus far the entire **Concurrent Model** was implemented in double precision. Running this model and saving the result on disk established a reference for evaluating the performance of the same model implemented in fixed point notation with various word lengths. In addition, determining the minimum and maximum values reached by each variable during the simulation established a good estimator for the optimal position of the decimal point of each variable.

To perform the procedure as described above Simulink provides a toolbox known as the **Fixed Point Tool**. To use this tool the same process as described by *Corless & Ananthan* [116] was followed. As a result, the **Concurrent Model** core as illustrated in Figure 154 could be converted from double precision notation into 16, 20 and 32 bit fixed point notations. Using these cores to assemble a 7x7 Array Processor and running the same simulation as described in paragraph 6.5.2 on the resulting **Concurrent Model** yielded the three load cases as illustrated in Figure 165, Figure 166 and Figure 167. These simulations revealed that the **Concurrent Model** yields an error of about 1 m/s for the *u* velocity component when implemented with 16 bit fixed point notation. In contrast the error is less than 0,025 m/s for the *u* velocity component when the **Concurrent Model** is implemented with 20 bit and 32 bit fixed point notations. Since the reduction in error from 20 bit fixed point notation to 32 bit fixed point notation was very small, the conclusion followed that a 20 bit fixed point notation is adequate for the pupose of the PRLS.

Note: *Although a word size of 18 bits may also produce acceptable results, this possibility has not been tested under this study due to time constraints. A word size of 18 bits could actually be a big advantage, since it is the default word size of the hardware multipliers of many FPGA types. Such a situation may well imply a sizeable reduction in the number of hardware multipliers required to implement the* ***Concurrent Model****.*

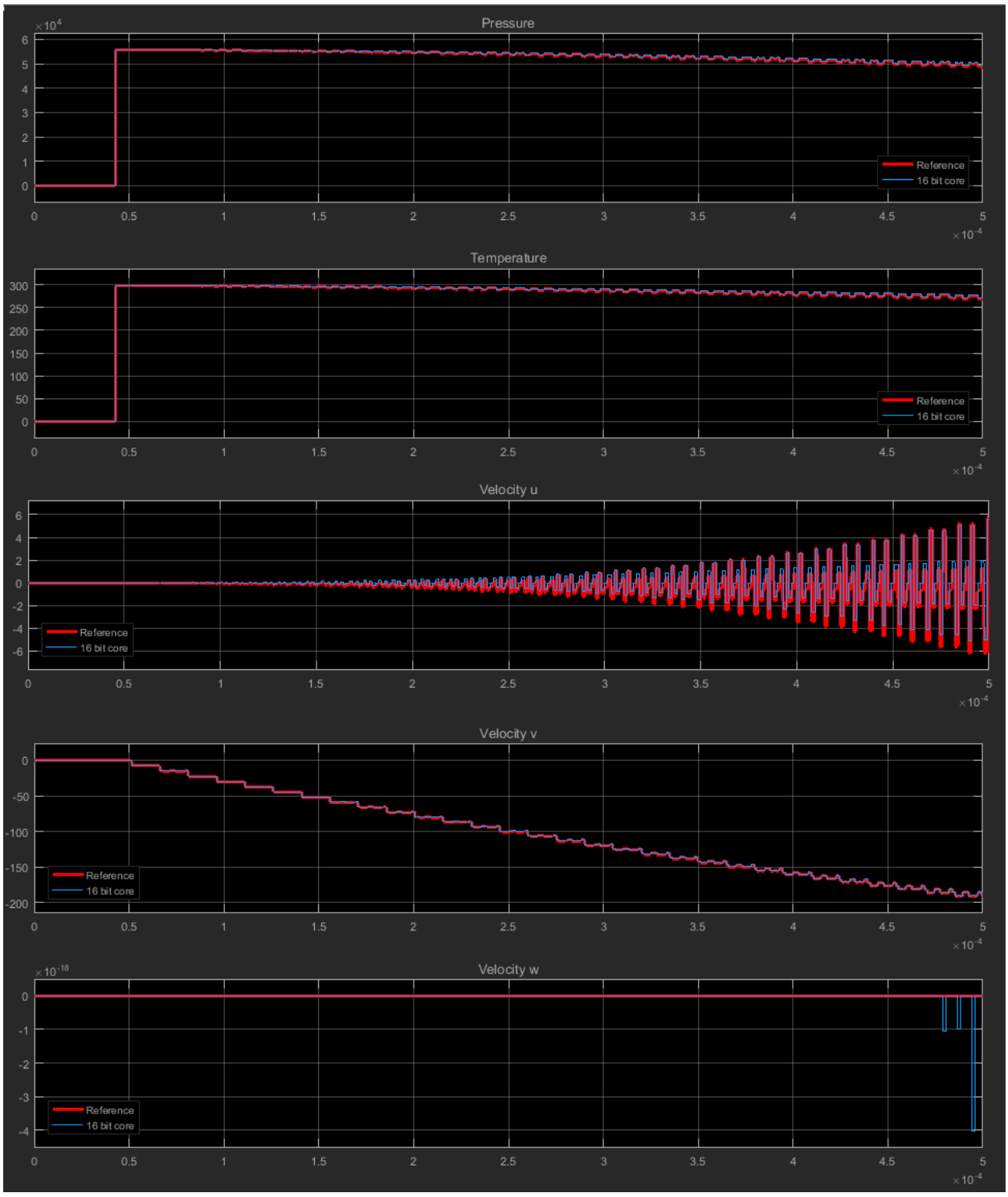


Figure 165. Performance comparison for 16 bit Concurrent Model, *RearNavierOut* bus [A Landman, 2017]

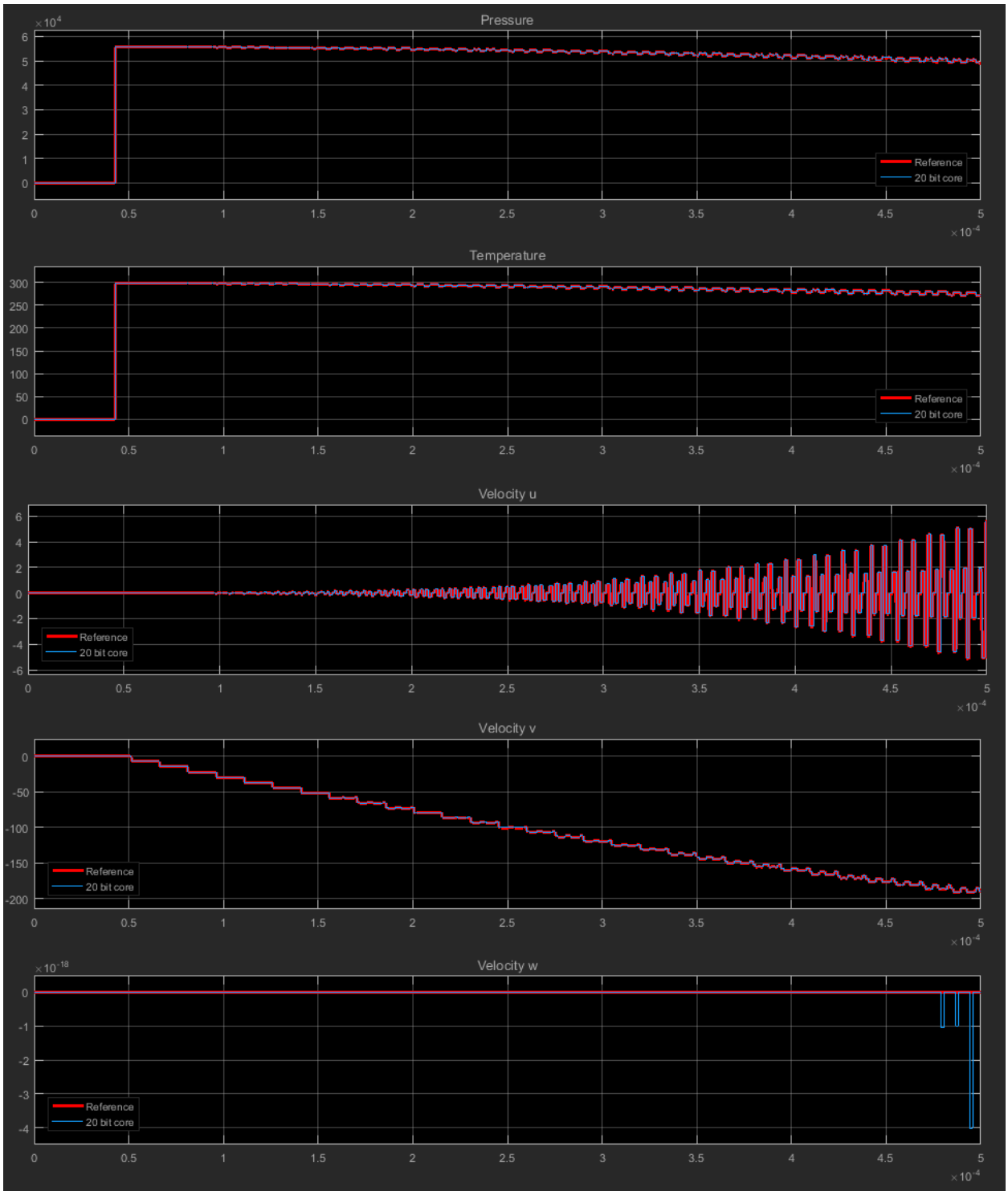


Figure 166. Performance comparison for 20 bit Concurrent Model, *RearNavierOut* bus [A Landman, 2017]

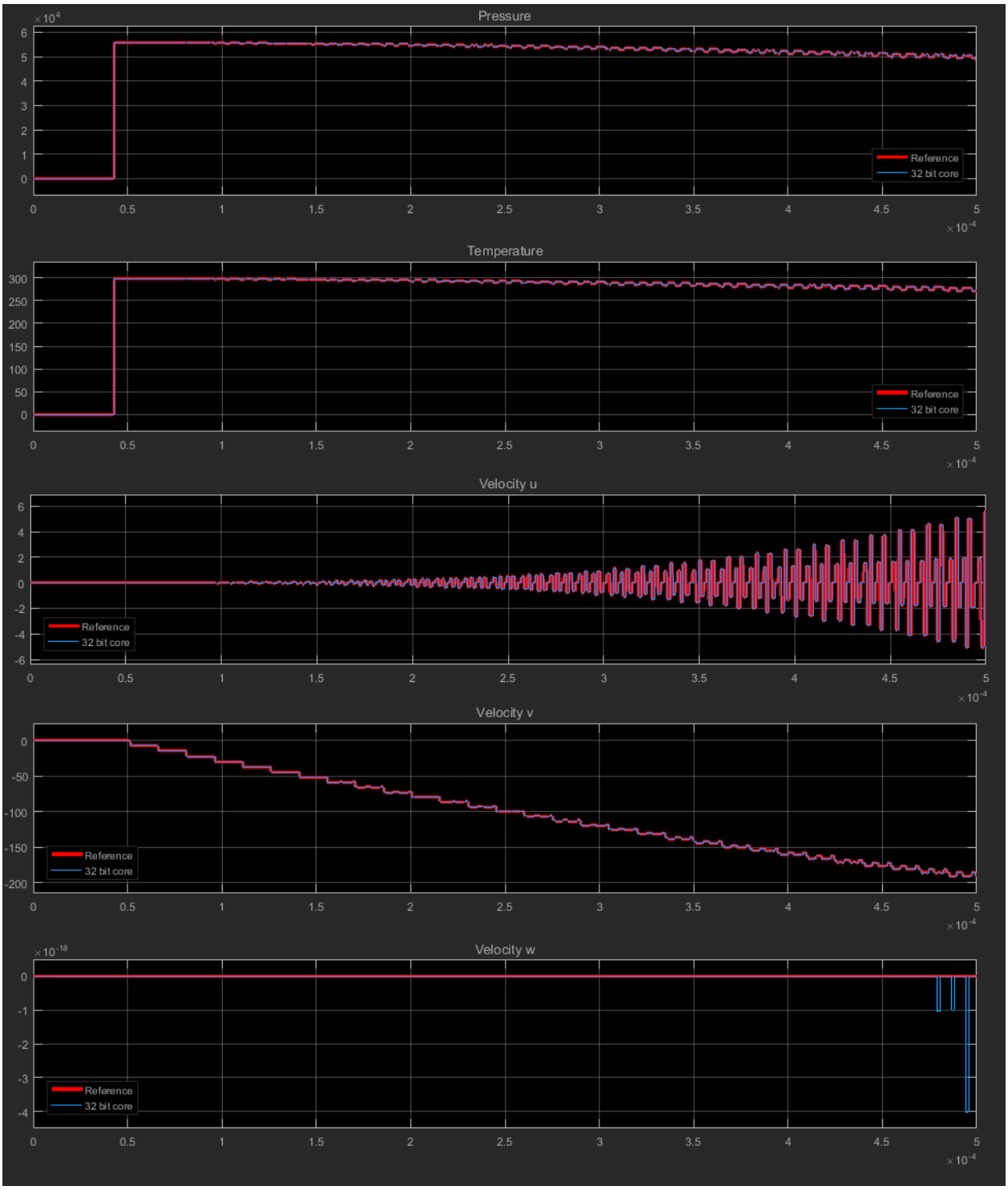


Figure 167. Performance comparison for 32 bit Concurrent Model, *RearNavierOut* bus [A Landman, 2017]

6.5.4 Embedding the Concurrent Model in FPGA

Implemented in 20 bit fixed point notation, the **Concurrent Model** core as described in paragraph 6.5 represented a working algorithm that could be transformed into Hardware Description Language (HDL) within the Simulink environment using the **HDL Coder**. The HDL files produced by the HDL Coder were subsequently fed into QUARTUS II for conversion into a binary file which can be downloaded into the FPGA via the Joint Test Action Group (JTAG) port of the device. During the process the HDL Coder also generated other characteristics of the circuit such as propagation delays, maximum clock frequency and resources used (e.g. number of multipliers and quantity of RAM) to implement the circuit. These are the parameters necessary to prove or disprove the Concurrent Hypothesis.

Note: *The **QUARTUS II** compiler as supplied by INTEL (previously ALTERA) was used for the purposes of this study. The same process can also be performed with the **VIVADO** compiler as supplied by XILINX. Although HDL compilers are also provided by other manufacturers , they have not been integrated with the HDL Coder of Simulink.*

Note: *The the HDL Coder works by generating a set of VHDL files that describe each of the blocks comprising the **Concurrent Model** core. It then creates a so-called Quartus project (suitably configured directory structure containing the VHDL files just created). Finally it activates the Quartus II compiler which compiles the project and generates the required binary file and associated performance data as described above. A large portion of these results are passed back to the HDL Coder for display or the entire data set can be viewed by invoking the QUARTUS II compiler directly.*

6.5.4.1 Resource use

Using the HDL Coder to convert the 20 bit **Concurrent Model** core into HDL revealed that the resulting circuit requires a total of 157 hardware multipliers to implement. Inspection of Figures 154, 157, 158, 159, 160, 161, 162, 163 and 164 shows that all of these multipliers are located within the **Filtered Equation** block. In addition, a multiplier requires much more gates (and hence is characterized by a much longer propagation delay) than things such as adders and buffers. The overall propagation delay of the 20 bit **Concurrent Model** core is therefore dominated by the **Filtered Equation** block which exhibits several critical paths, each containing many multipliers and adders in series.

In order to achieve an iteration rate of 10 kHz minimum it is therefore necessary for all of these multipliers to be implemented in hardware. The objective must be to select a device which contains the largest number of hardware multipliers possible with just enough fabric, RAM and serial I/O to implement the DFBD as described in paragraph 6.5. Devices containing integral processors and other features such as timers, ADC convertors, DAC convertors and external memory controllers must be avoided since these features would merely translate into wasted FPGA resources.

6.5.4.2 Performance evaluation

The worst-case Propagation Delays of the 20 bit **Concurrent Model** core when implemented on the candidate FPGA devices of Table 7 were calculated with the HDL Coder and the results listed in Table 8. Also shown in the same table are the fractions of available ALM, RAM and multiplier resources used to implement the 20 bit **Concurrent Model** core on these devices.

Table 8. Resources used to implement 20 bit Concurrent Model core on candidate FPGA devices [A Landman, 2017]

	CYCLONE V	ARRIA 10	STRATIX V	STRATIX 10
Device	5CEFA9F23C7	10AS066H2F34E1SG	5SGSMD8K1F40C2	GX2800
Critical path propagation delay	67,8 ns	71,1 ns	31,5 ns	31,5 ns (2)
Max (1) iteration rate	29,4 kHz	28,0 kHz	63,4 kHz	63,4 kHz (2)
ALM use	3%	1%	1%	0,28% (2)
RAM use	2%	1%	1%	0,22% (2)
Multiplier use	45%	9%	8%	2,72% (2)

Note (1) : For a Flight Box containing 500x500x500 cells

Note ‡ : Estimated figure (compiler for STRATIX 10 was not available at the time of this study)

The conclusions that can be drawn from Table 8 are as follows:

a) The required iteration rate is easily met with any of the candidate devices and can be as high as 63,4 kHz when implemented on device GX2800.

b) The Multiplier to Fabric Usage Ratio is very low, ranging from 15:1 for mid-range devices such as the CYCLONE V device family to 10:1 for top-end devices such as the STRATIX 10 device family. The implication of this is that although contemporary FPGA devices can be used to achieve the desired iteration rate of 10 kHz through extreme parallelism, the solution is very inefficient in terms of resource use.

c) Should device 5CEFA9F23C7 of the CYCLONE V family be used to implement the **Concurrent Model** on a 500x500 Planar Array Processor it would require a total of 125000 devices, forming a sheet 7,4 x 7,4 meter in size. This is an order of magnitude more than the total exterior surface area of the Precision Rocket Launcher. Should device GX2800 of the STRATIX 10 family be used it would require a 83x83 array of devices (i.e. 6889 in total), forming a sheet 4,5 x 3,7 meter in size. Even this solution would be much more than the total exterior surface area of the Precision Rocket Launcher.

6.6 Summary (Develop General Theories)

This chapter was used to perform the sixth Scientific Method Cycle of a family of nested Scientific Method Cycles that yield the concept of a Precision Rocket Launching System. Chapter 4 derived a **Gas Flow Model** that is suitable for solving the Navier Stokes equation for a given flight box and implementation with massive parallel schemes. Chapter 5 researched mechanisms to control the **Gas Flow Model** and a Planar Array Processor with associated re-

use for implementing the resulting model in a massively parallel circuit. This chapter developed the first iteration of such a circuit within the Simulink development environment, expressed the circuit in HDL language and compiled it for download into a selection of mid-range and top-end FPGA devices.

The observational evidence included the facts that the estimated computational load of the **Gas Flow Model** amounts to 0,601 petaflop while Chapter 5 showed the maximum theoretical throughput of the Planar Array Processor to be 5,75 petaflop based on manufacturers data. Other observational evidence included a range of FPGA devices that might be suitable for implementing the Array Processor, identification of data types that can be used for implementing the **Gas Flow Model**, description of the Serial Peripheral Interface (SPI) mechanism and how it can be used as Nearest Neighbour Interface between adjacent FPGA devices and the role of determinism and predictability to certify safety critical software. Finally, various methodologies and tools for developing such a large circuit were presented, including the use of Hardware Development Language (HDL), Simulink as Integrated Development Environment (IDE) and the Quartus II compiler.

Questions were asked as to the possibility of embedding the **Downwash Model** in a bespoke circuit comprising quadrillions of transistors that all operate concurrently in a harsh environment. Observational evidence that suggest it may be possible was presented, including extreme miniaturization of electronics over the past century, developments in production and packaging techniques of contemporary electronics and the ability to simulate and verify the behaviour of the circuit beforehand.

These questions led to formulation of the Concurrent Hypothesis which states that a literal implementation of the **Concurrent Model** at the re-use operating point implied by the Planar Array Processor of the Precision Rocket Launcher should be possible and that it should be possible to drive this circuit with a central state machine such that the housekeeping limit of Amdahl can be avoided. The Concurrent Hypothesis also stated that the circuit can be implemented in fixed point notation, that it can run at an iteration rate in excess of 10 kHz and that it can be designed and tested using Hardware Development Language within the Simulink Integrated Development Environment.

Generating a literal circuit implementation of the **Concurrent Model** running on the Planar Array Processor of the Precision Rocket Launcher showed that while a 20 bit fixed bit notation yields acceptable accuracy and more than enough speed the the resulting circuit has a Multiplier to Fabric Ratio that is very low, ranging from 1:15 for mid-range devices such as the CYCLONE V device family to 1:10 for top-end devices such as the STRATIX 10 device family. In this type of design the number of cores that can be fitted into a given FPGA device is completely dominated by the number of hardware multipliers present in the device.

The design also showed that in this type of design the number of FPGA devices required exceeds the spatial limits of the Precision Rocket Launcher with almost an order of magnitude. For optimal usage of resources this type of design will require a FPGA with 10 times the number of hardware multipliers currently contained in top-end FPGA's such as device GX2800 from the STRATIX 10 family. No commercially available FPGA device that complies with this requirement could be found as at 2018.

The main contribution of this chapter was establishment of the fact that the level of reuse as implied by a literal circuit implementation of the **Concurrent Model** running on the Planar Array Processor of the Precision Rocket Launcher using contemporary FPGA devices leads to a solution that can solve the Taylored Navier Stokes equation at a rate in excess of 10 kHz but has physical dimensions that exceed the physical limits of the Precision Rocket Launcher, proving the Concurrent Hypothesis untrue. Some further increase in the level of re-use (at the expense of iteration speed) is therefore required to meet the performance levels required by the PRLS Hypothesis. Finding such a solution is the objective of the next Chapter.

7 The Pipeline Model (Cycle 7)

"*The best way to predict the future is to create it*". Peter Drucker.

This chapter introduces the seventh Scientific Method Cycle of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. As shown in Chapter 2 the downwash of the helicopter must be calculated with good accuracy and in real time for the PRLS concept to work. To this effect, chapters 4, 5 and 6 were devoted towards developing a suitable algorithm for calculating the downwash flow field and conceiving a Precision Rocket Launcher containing a Planar Array Processor and associated circuit for performing the calculation in real time. That effort showed that if contemporary FPGA devices are used the level of re-use as implied by a Planar Array Processor is low enough to achieve the required iteration rate of 10 kHz but too low to ensure that the circuit can fit into the physical limits of the Precision Rocket Launcher. The reason for this problem was found to be the liberal use of hardware multipliers that characterizes the **Concurrent Model**.

Figure 168. Exterior view of Stratix 10 BGA package [Intel: 2016]

The purpose of this chapter is to modify the **Concurrent Model** in such a way as to decrease the use of hardware multipliers to a point where the new circuit can fit within the resources of the Precision Rocket Launcher. Questions are asked as to how this can best be done, leading to the Pipeline Hypothesis which states that it can be achieved by increasing the level of re-use of the **Concurrent Model** at the expense of iteration rate.

The Pipeline Hypothesis is tested by re-arranging the the **Concurrent Model** into a systolic (pipeline) configuration through firmware changes, avoiding hardware changes to the Precision Rocket Launcher and leading to a solution referred to as the **Pipeline Model** – a solution that has both reasonable speed and practical resource requirements given the limits of contemporary electronics, in particular device GX2800 as illustrated in Figure 168.

The main contribution of this chapter is proof that as at the year 2018 there is in existence at least one point in the re-use workspace where a bespoke algorithm can calculate with high accuracy the downwash flow field of a helicopter in real time and within the physical limits imposed by an airborne application. Apart from the PRLS this bespoke arrangement can be applied to other applications with similar demanding limitations.

At philosophical level conception of the **Pipeline Model** can be considered as just another milestone in the evolution of computational capability of man as reflected by contemporary electronics and observed by *Moore* [84] and *Kurzweil* [85]. The observational evidence would seem to indicate a world moving at ever increasing speed towards a state where re-use and the cost of extreme parallelism tends towards zero. Where will this trend stop? An apparent stop was indicated by *Amdahl*, only to be disproven by *Gustafson* a few years later. Our own research shows that extreme levels of parallelism is possible provided centralized control can be minimized – as is the case for many systems in nature. Under these conditions the performance of the system is throttled by the local propagation of information – propagation delay in the case of contemporary electronics and speed of light in case of space.

7.1 Too many multipliers (Make observations)

In the previous chapter it was shown that the **Downwash Model** can be solved at a rate as high as 63,4 kHz when the Planar Array Processor is implemented with contemporary FPGA devices and configured as a literal implementation of Equation 105 and the process as described in Chapter 4.

Although the concept of calculating the downwash of a helicopter in real time using on-board equipment has therefore been proven as viable, it was also shown that the **Concurrent Model** contains two critical flaws. The first flaw is that it requires too much resources to fit into the physical constraints of the Precision Rocket Launcher as described in Chapter 5. The second flaw is that it results in a serious mismatch between hardware multipliers and FPGA fabric. A typical ratio would be 10:1 when a single **Concurrent Model** core is implemented on a top end FPGA. Such a mismatch is not desirable since it leads to unused FPGA fabric which results in elevation of cost. This flaw is strongly amplified by the fact that each Precision Rocket Launcher will contain hundreds of FPGA devices, resulting in so high a cost of implementation that the Precision Rocket Launcher may become unaffordable.

It is evident that a literal translation of the process as described in Chapter 4 into an FPGA circuit results in too many hardware multipliers. In other words, an overuse of a scarce resource. As described in paragraph 5.5.1 the de-facto solution to this problem is to increase the level of re-use. In this case it would be desirable to implement such a change through firmware changes (i.e. no hardware changes).

7.2 A systolic process (Think of interesting questions)

Inspection of Equation 105 shows that it can be decomposed into seven discrete components as described in paragraph 6.5.1. In the **Concurrent Model** core these components are arranged like a “transmission line” as illustrated in Figure 157. In other words, data follows a sequential path through the transmission line with additions along the way similar to blood flows joining along an artery. In the **Concurrent Model** the entire sequence must complete during each minor cycle, so much of the circuit is actually inactive most of the time. Each minor cycle generates data which travels like a “wave” through the transmission line. This leads to a question - can the data of successive minor cycles not be fed into this transmission line in rapid succession, such that the transmission line contains multiple waves of data and more of the circuit can be active at any given point in time?

Note: *The fact that the* ***Concurrent Model*** *reacts like a transmission line containing peaks and lulls of data explains the apparent paradox resulting from the theoretical maximum throughput of 5,75 petaflop of the Planar Array Processor as derived in paragraph 5.2.3 versus the actual throughput of 0,601 petaflop as derived in paragraph 7.6.1.*

The arrangement as described above is similar to that of a one-dimensional Systolic Processor which can be implemented using contemporary electronics as described by *Baur, Gläß, Klefenz, Long & Männer* [120]. If such an arrangement can be contrived, the minor cycles can be made to overlap (without increasing the time duration of the major cycle) allowing more time so that each process of the transmission line can be implemented with a sequence of operations centred around a single hardware multiplier.

7.3 The Pipeline Hypothesis **(Formulate hypothesis)**

Considering all the observational evidence as presented above we now postulate that it should be possible to design a bespoke circuit referred to as the **Pipeline Model** that implements each cell of the **Downwash Model** as a one-dimensional systolic processor (pipeline) which should cause an increase in the level of re-use as implied by the basic Planar Array Processor. This increase in re-use should enable reduction of hardware multipliers required, such that the overall circuit will fit within the resources of the Precision Rocket Launcher as defined in Chapter 5. It should be possible to coordinate all of these cells with a central state machine that performs a known and repeating state transition sequence that avoids the housekeeping limit of Amdahl. It should also be possible to implement this circuit in fixed point notation and run it at an iteration rate in excess of 10 kHz. Finally, it should be possible to design and test this circuit using Hardware Development Language within the Simulink Integrated Development Environment.

7.4 Pipeline predictions **(Develop testable predictions)**

If the Pipeline Hypothesis is true the **Pipeline Model** will be capable of estimating the downwash flow field of a helicopter within a 100m x 100m x 100m flight box at a temporal resolution smaller than 0.1 millisecond, a spatial resolution smaller than 200 mm and within the resources and constraints of the Precision Rocket Launcher.

7.5 Conceiving the Pipeline **(Gather data to test predictions)**

To conceive a bespoke circuit for the **Pipeline Model** core note has to be taken that such a circuit corresponds in various aspects with systolic processors of which a wide range of examples are available in the open literature. Like a systolic processor the output of a given block should be sampled at the start of a clock cycle and held constant to serve as (constant) input to the next block during the rest of the clock cycle. However, there are also fundamental differences. Where the transfer function for each cell of a systolic processor is of the form *A=B+C*D* the transfer functions of the blocks comprising the **Pipeline Model** core are complex and different from each other.

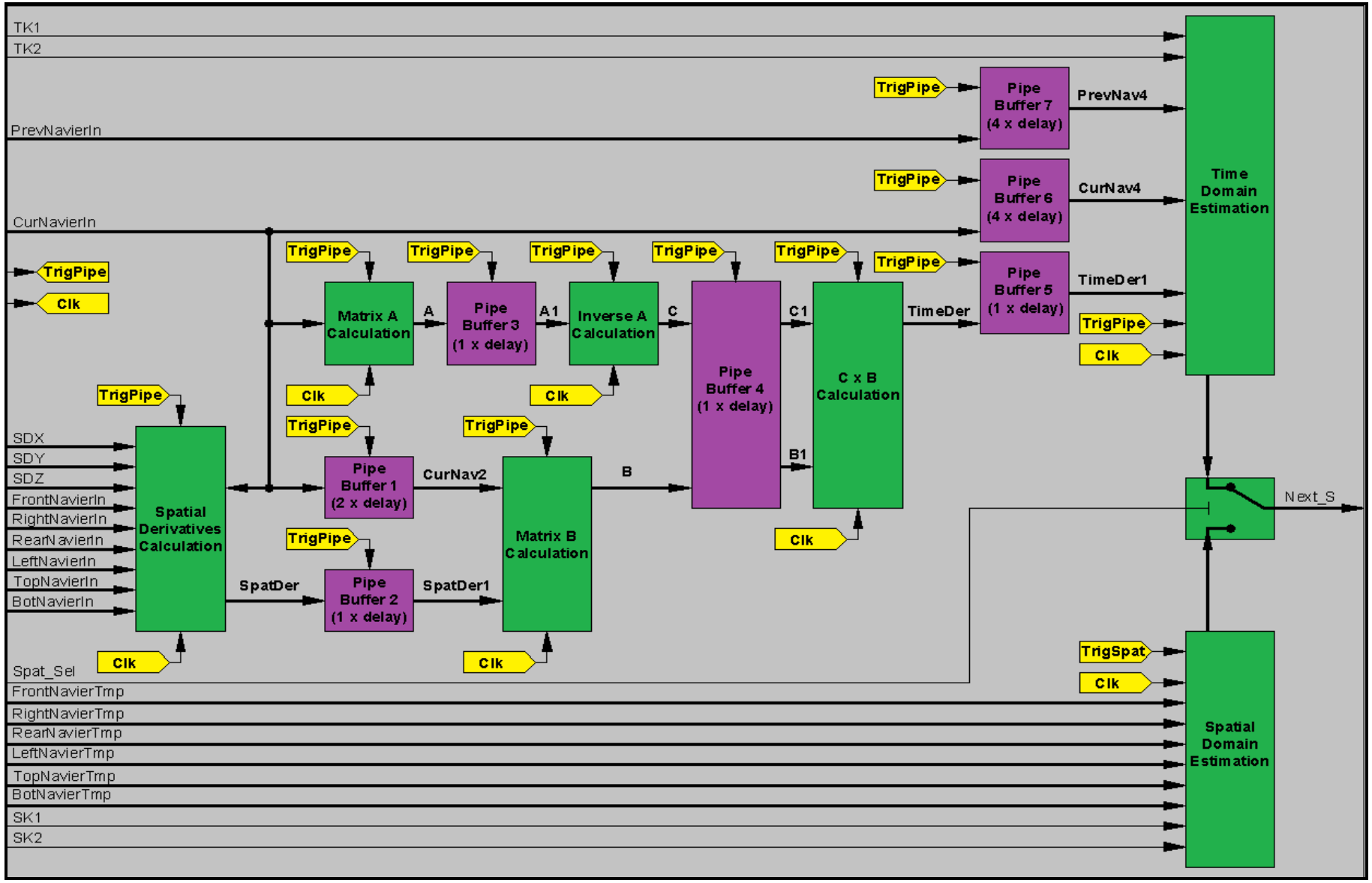


Figure 169. DFBD for the Pipeline Model core [A Landman, 2018]

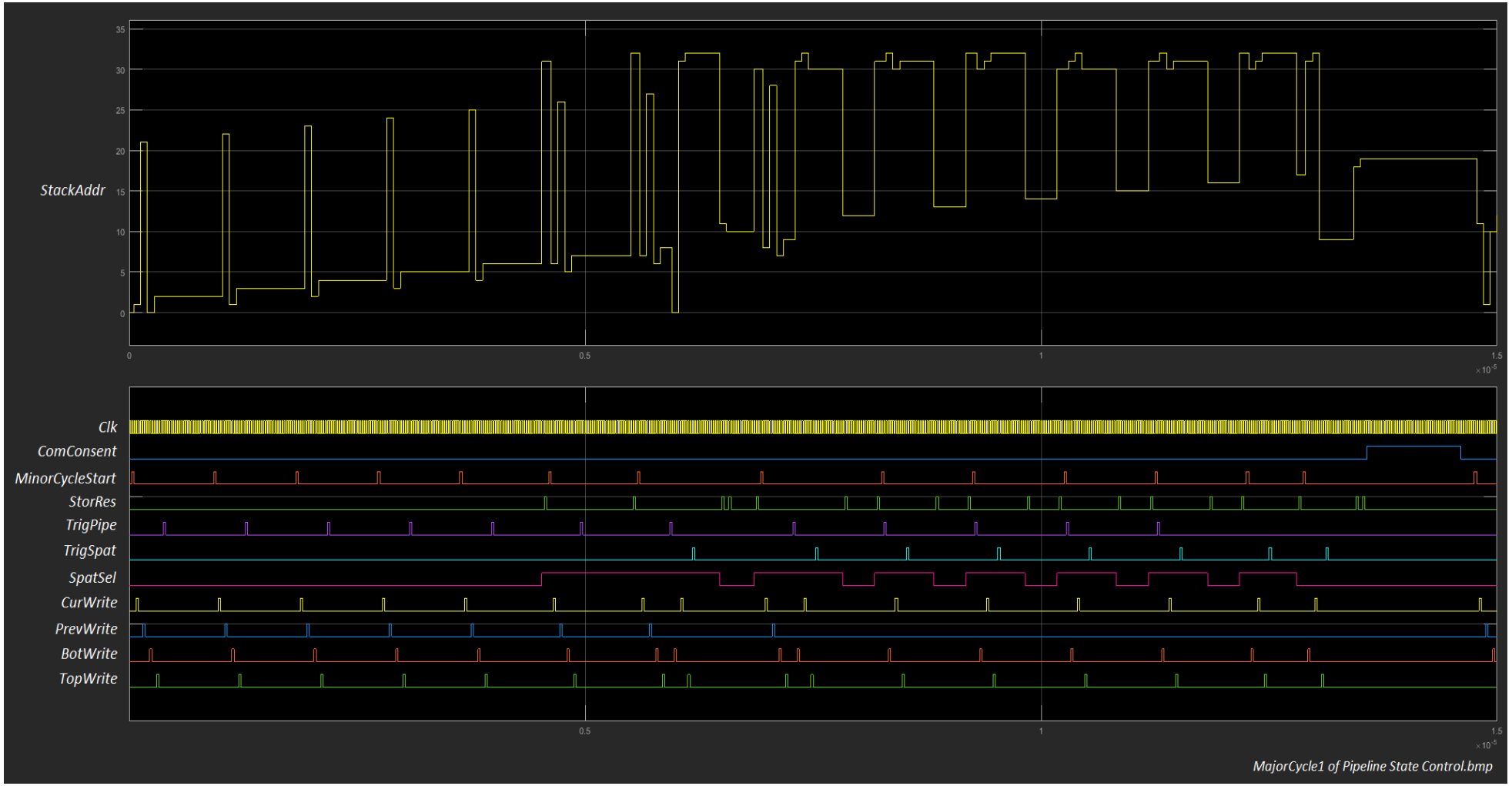


Figure 170. Example of signals from Navier Control block in Pipeline Model with 10x10x10 Flight Box [A Landman, 2016]

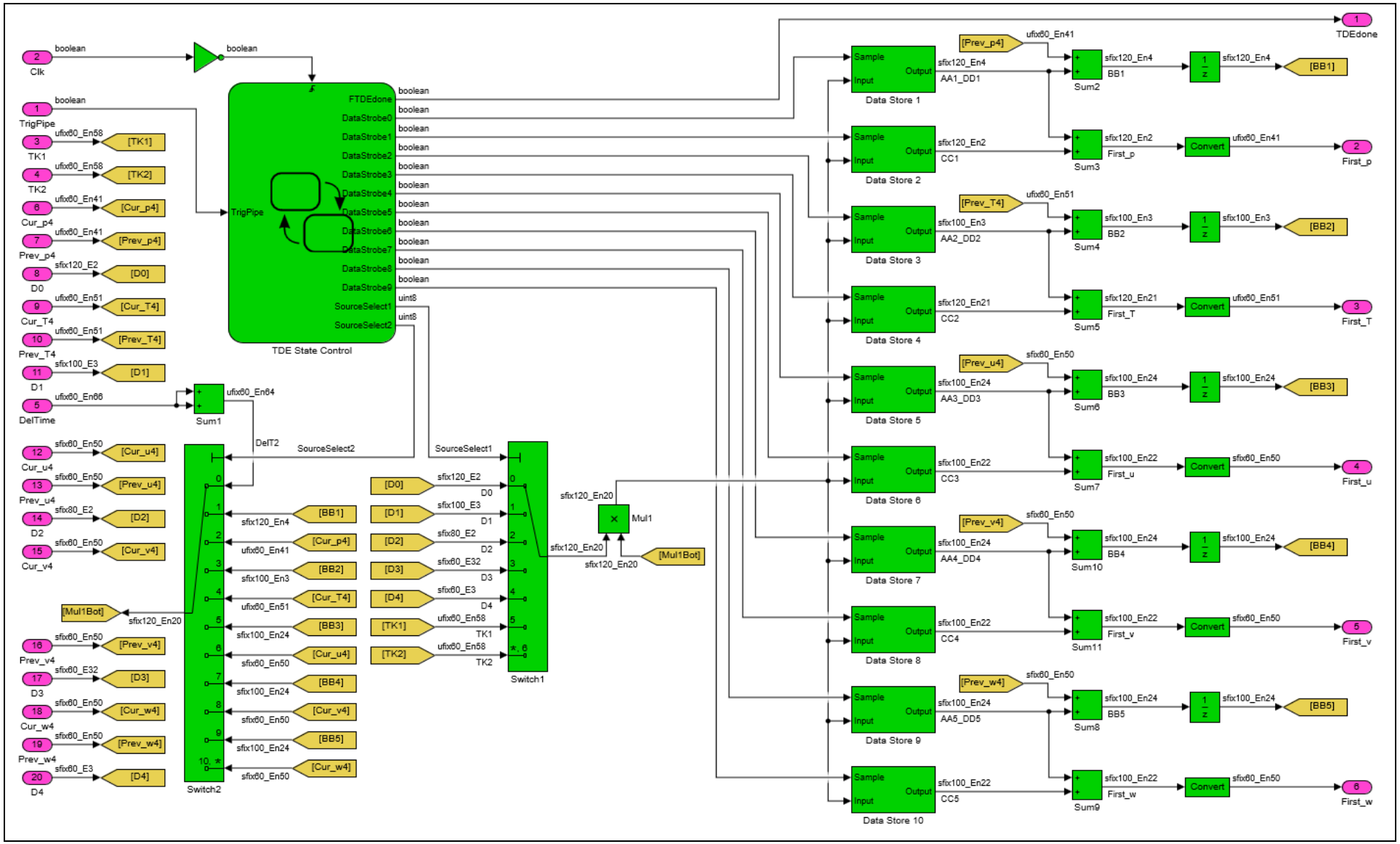


Figure 171. Pseudo-processor for the Time Domain Estimation block [A Landman, 2016]

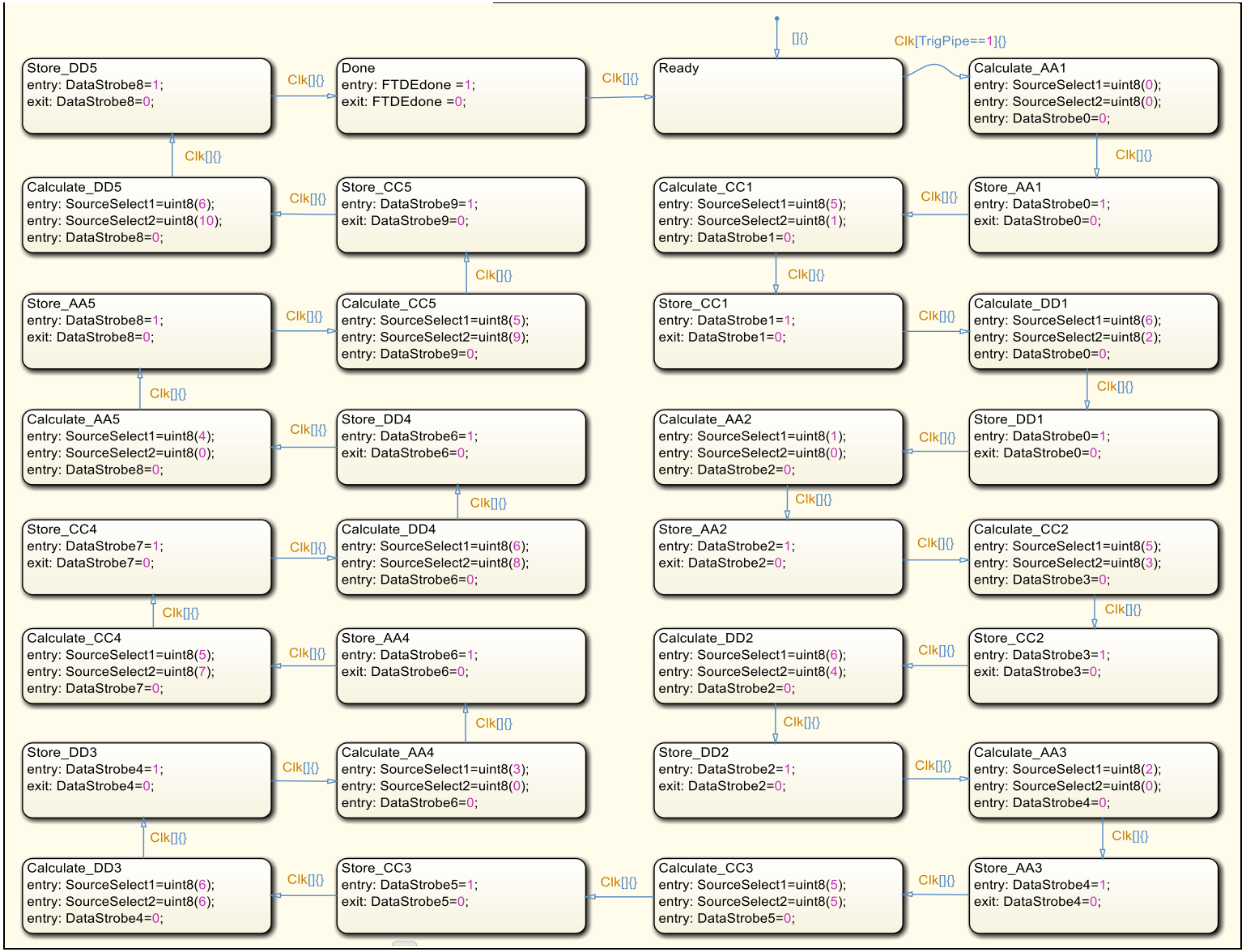


Figure 172. State transition diagram for the TDE State Control block [A Landman, 2016]

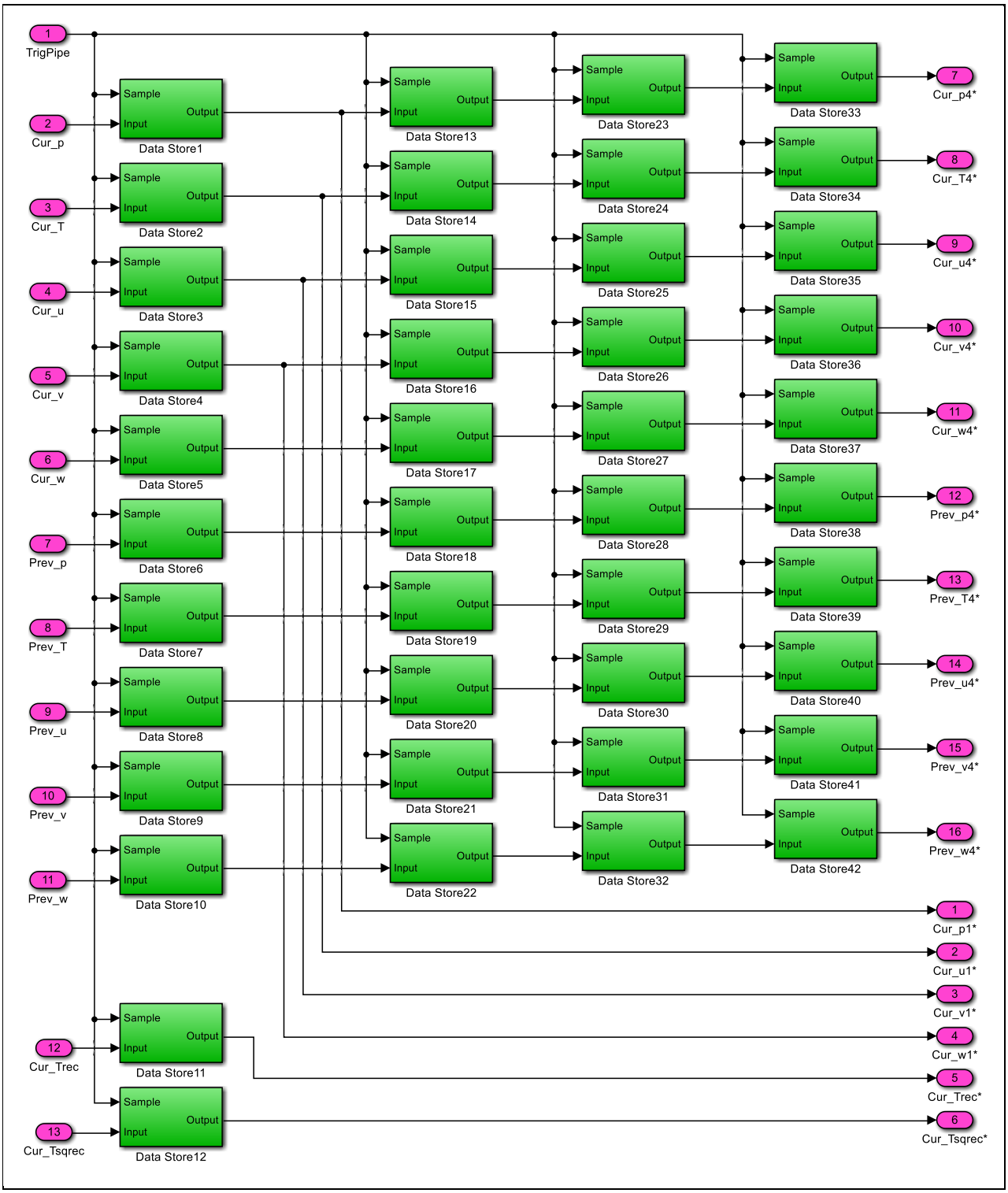


Figure 173. Pipe Buffer with taps at 1 and 4 clock pulse delays [A Landman, 2016]

Using the factors as identified in paragraph 7.2 to conceive a systolic implementation of the **Concurrent Model** led to the DFBD as illustrated in Figure 169. In this arrangement a number of sample and hold registers referred to as **pipe buffers** were inserted at strategic positions between the original blocks of the **Concurrent Model** core as illustrated in Figure 157 showing pipe buffers in purple. This arrangement makes it possible to shift the data sets of four consecutive minor cycles through the **Pipeline Model** core in a systolic manner so that they can be processed concurrently. More of the circuit thus performs useful work throughout the minor

cycle, allowing some parts of the circuit to operating sequentially in order to increase the level of re-use employed in the circuit.

After being triggered by a *TrigPipe* pulse each pipe buffer samples its input and shifts it one position along a serial arrangement of buffers (pipe) as illustrated by the example of Figure 173. Tapping the data at different points along the pipe allows the input to be delayed with different amounts of trigger pulses. This is necessary to ensure related data sets arrive at an associated block at the same time.

Functionally, the green blocks of Figure 169 are the same blocks as used in the **Concurrent Model** core of Figure 157. These blocks are triggered by the *TrigPipe* and *TrigSpat* pulses which cause them to perform a sequence of actions (pseudo-program) in phase with the *Clk* signal. Where the execution times of these blocks in the **Concurrent Model** core were determined by propagation delay, they are now determined by the frequency of the *Clk* signal

Apart from a change in the state transition logic of the **Navier Control** block and the change in internal organization of individual cores, the overall layout of the **Pipeline Model** remains essentially the same as that of the **Concurrent Model** as illustrated in Figure 153. The **Navier Control** block requires modification because the blocks of the **Pipeline Model** must be coordinated, albeit with a pre-determined sequence to avoid the performance limitation of Amdahl.

For the **Pipeline Model** core the green blocks of Figure 169 are implemented as circuits arranged around a state machine that controls a single hardware multiplier. As example, the DFBD and state transition diagram of the **Time Domain Estimation** block are illustrated in Figure 171 and Figure 172. Since the state transitions of the **TDE State Control** block are triggered by the *Clk* signal, the **Time Domain Estimation** block of the **Pipeline Model** core requires a finite time to execute. This same reasoning also applies to the other pseudo-processors of the **Pipeline Model** core, resulting in the execution times as illustrated in Figure 174.

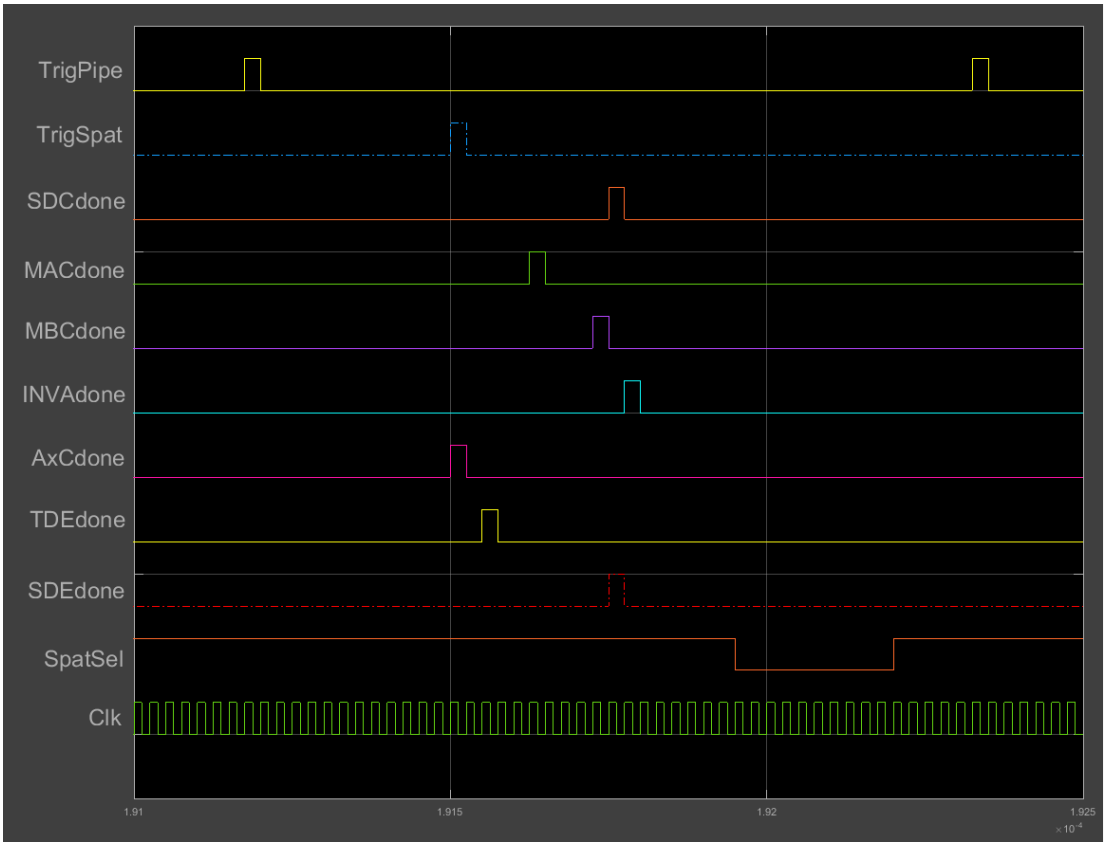


Figure 174. Acknowledge sequence of the Pipe following *TrigPipe* pulse [A Landman, 2017]

Since the execution time of the **Spatial Domain Estimation** block of the **Pipeline Model** core is relatively short, spatial filtering of the solution could be performed in parallel to calculation of the initial solution in the time domain.

7.5.1 Verification of the Pipeline Model

Correct operation of the **Pipeline Model** was verified by comparing the video sequence (movie) as generated with the Big Box experiment of paragraph 4.5.8 with the same video sequence (movie) as generated by the **Pipeline Model**. Both models produced the same result.

The biggest problem of this test was extremely long simulation times. Whereas the Matlab script developed during the Big Box experiment required only a few minutes to model about 500 msec of real world operation, the **Pipeline Model** required weeks to do the same.

As for the **Pipeline Model** the objective here was not to produce an exact result, but to confirm that the **Pipeline Model** executes the same mathematics as the Matlab script developed under paragraph 4.5.8. This is all that is required at this stage, since the objective was to determine if Equation 105 embedded in the fabric of an array of FPGA's can be clocked fast enough to yield an overall iteration rate of 10kHz.

7.5.2 Converting the Pipeline Model to Fixed Point

Having developed the **Pipeline Model** in double precision the **Pipeline Model** now had to be converted into a suitable fixed point notation, the objective being to maximize clock speed and to minimize the amount of FPGA recources needed.

Note that "suitable fixed point notation" in this context means to represent each variable of the system with enough bits and decimal point located such as to yield both enough dynamic range (to prevent overflow) and enough resolution (to achieve suitable accuracy and avoid quantization noise). This problem represents a discrete branch of mathematics known as **Interval Arithmetic** [121]. The literature contains many papers on this topic, many of which describe some or other technique (often using a so-called "compiler") to perform the conversion for some or other specialist area. Searching for an alternative method that requires less time to perform, *Deest, Yuki, Sentieys & Derrien* [122] determined the effects of converting an embedded algorithm from floating-point notation to integer notation by using convolution to determine an impulse response that is representative of the embedded code. Most of these techniques assume a fixed word length throughout the converted algorithm similar to the Fixed Point Tool of Simulink. *Lin, Talathi & Annapureddy* [123] found that different word lengths can result in resource savings up to 20% with no reduction in accuracy (similar to our own findings). Many of the papers define a form of Signal to Noise Ratio (SNR) as criteria to determine word length and decimal point position for each variable in the system.

For the purposes of this study an attempt was made to use the Fixed Point Tool of Simulink to convert the **Pipeline Model** configured for an array of 10x10 cores into fixed point notation. In doing so excessively long simulation times were encountered (using Qosmio gaming laptop with CORE i7 running at 2.4 GHz). For example, with the **Pipeline Model** configured for an array of 10x10 cores the simulation ran for almost two days in Accelerator mode to collect a set of double-precision reference data spanning a period of only 0,7 msec.

*Note: To generate a reference data set large enough to contain all the min/max values of the system under test with high enough probability would require running the **Pipeline Model** for years. In practice the fixed point notations as proposed by the Fixed Point tool of Simulink may well imply limits that are too small for representing the numbers that would occur in a real system. To ultimately eliminate this problem an actual Planar Array*

Processor operating under real world conditions would have to be monitored in real time and FPGA adjustments made where overflow conditions do occur.

Having to spend several days running the **Pipeline Model** to collect reference data spanning 0,7 msec we concluded that running this model on a standard laptop was pushing the Simulink operational boundary to the brink. Correcting even minute errors in the model under these conditions became major time issues. At this point our scheme for scaling large models with the Fixed Point Tool went astray when the tool crashed after 3 days of processing. This was due to an attempt by the tool to write to memory beyond the 32 Gbyte limit of the work station (probably due to some or other limit in the program that was exceeded due to the sheer size of the simulation). This bug in the Fixed Point Tool forced us to revert to a second operational mode of the Fixed Point Tool called **Rate Range Analysis** – a different name for Interval Arithmetic**.** Unfortunately this mode of the Fixed Point Tool also failed when the program viewed a signal as a block which it could not find, causing the program to terminate.

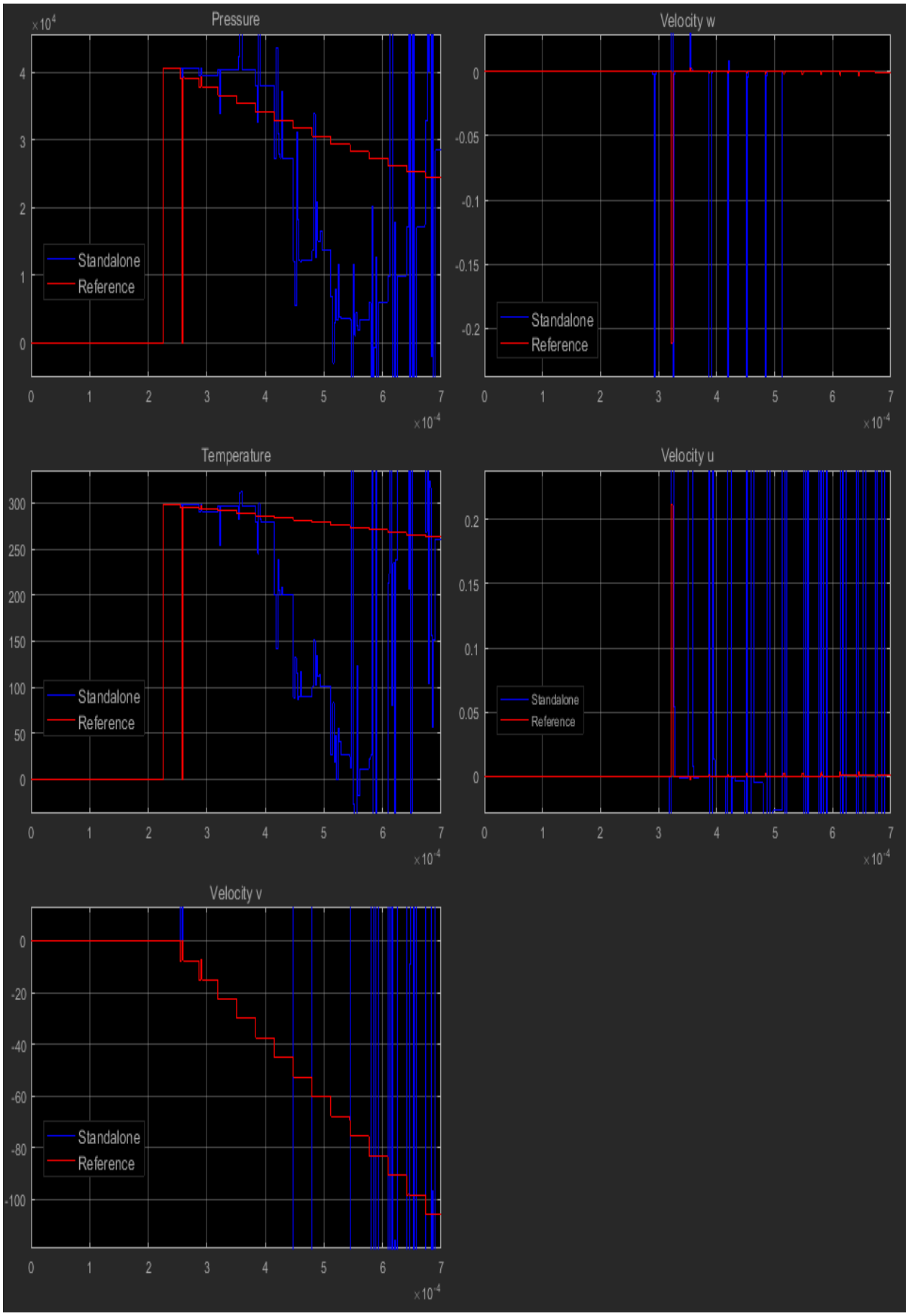


Figure 175. Response of Pipeline Model: sub-optimal word lengths [A Landman, 2017]

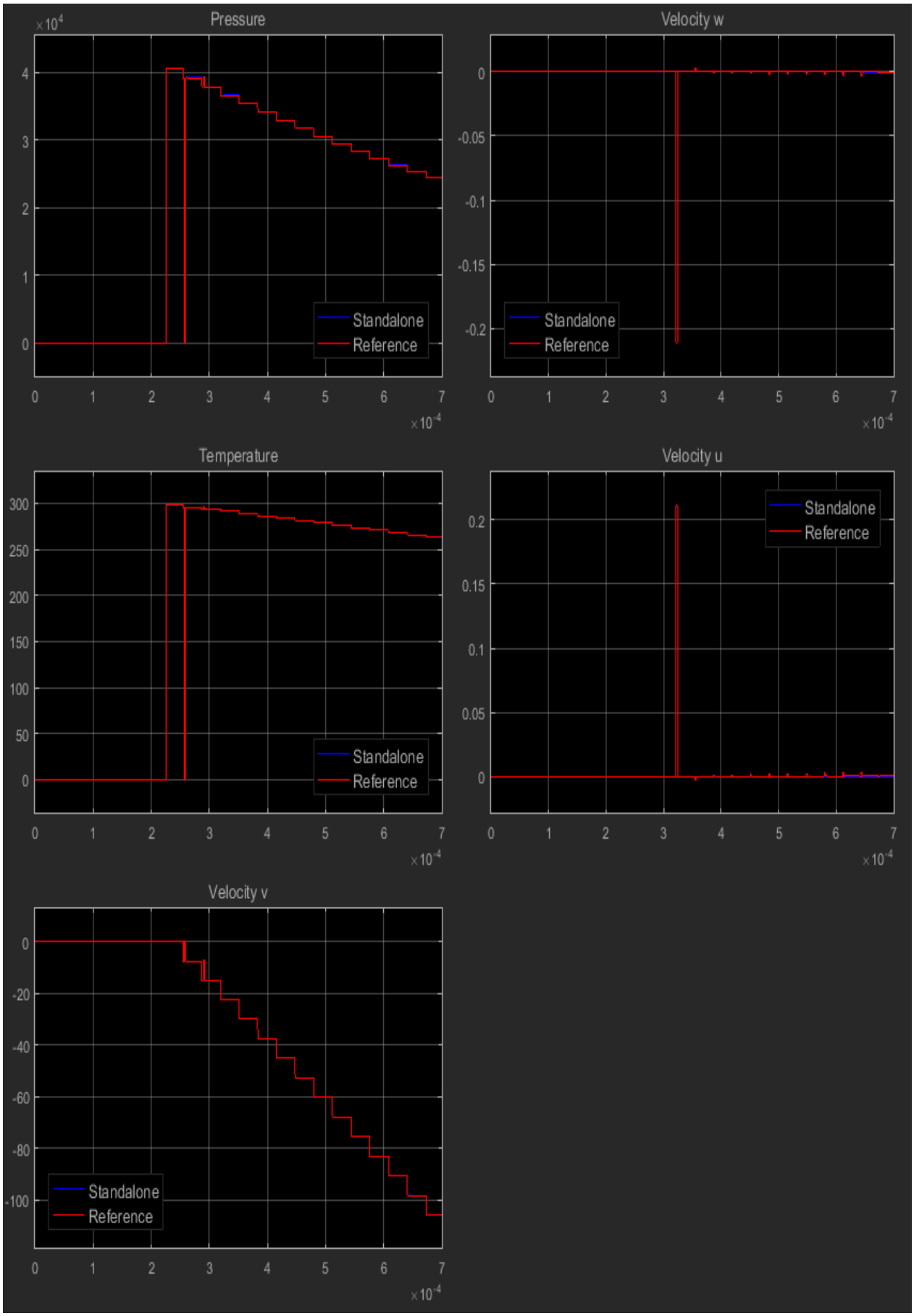


Figure 176. Response of Pipeline Model: sweet spot word lengths [A Landman, 2017]

Having depleted all options to automatically scale the **Pipeline Model** using the Fixed Point Tool the **Pipeline Model** had to be scaled by hand. This was done by making notation adjustments for each variable (one by one) and running the **Pipeline Model** to compare the result of the change with the double-precision reference result (i.e. a type of SNR analysis). To do this the resolution of each signal was gradually adjusted, resulting in the typical responses (Nearest Neighbour signals) as illustrated in Figure 175 and Figure 176. Here the first figure shows the response of the **Pipeline Model** for a signal represented with insufficient resolution and the second figure the response of the **Pipeline Model** for the same signal represented with just enough resolution (the so-called "sweet spot").

Note: *Similar to the results as described in [123] manual scaling of the **Pipeline Model** as conducted under this study also resulted in various word lengths. In fact, it showed that using the same word length throughout the **Pipeline Model** would have resulted in a in multiplier and gate count penalty much larger than the 20% as reported by Lin, Talathi & Annapureddy* [123].

One of the effects noticed during manual scaling of the **Pipeline Model** core was that the resolution requirements of blocks increased moving from inputs to outputs. Typically, many signals of the **Spatial Derivatives Calculation** block could be represented with as few as 20 bits while some signals of the **Time Domain Estimation** block had to be represented with 120 bits. This effect is contrary to what the Butterfly Effect as discovered by *Lorenz* [25] would seem to suggest. According to this effect the paths in state space (i.e. histories) of the various solutions should rapidly diverge if the initial conditions are only slightly different (in this case changed by different quantization noise). This apparent paradox is probably due to the fact that the blocks at centre of the core are responsible for calculating the inverse of a matrix and predicting the future using partial derivatives (which tend to amplify noise). This explanation is supported by the fact that the need for wide word widths starts to decrease in the **Time Domain Estimation** block.

Another effect noticed was that the performance of the **Pipeline Model** is dominated by blocks referred to as "cornerstone blocks" for the purpose of this study. As a general rule, all the multiplier blocks and inverse calculating blocks resort under this category.

7.5.3 Embedding the Pipeline Model in FPGA

The **Pipeline Model** configured with sweet spot fixed point notations as described in paragraph 7.5.2 now represented a working algorithm that could be transformed into Hardware Description Language within the Simulink environment using the **HDL Coder**. The HDL files produced by the HDL Coder were subsequently fed into QUARTUS II for conversion into a binary file which can be downloaded into the FPGA via the Joint Test Action Group (JTAG) port of each FPGA device. During the process the HDL Coder also generated other characteristics of the circuit such as propagation delays, maximum clock frequency and resources used (e.g. number of multipliers and quantity of RAM) to implement the circuit. These are the parameters necessary to prove or disprove the Pipeline Hypothesis.

7.5.3.1 Resource use

Using the HDL Coder to translate the **Pipeline Model** core into HDL showed that the **Pipeline Model** requires only 10 multipliers per core versus the **Concurrent Model** which requires 157 multipliers per core. This comes at a price since the need for other resources such as adders, registers and multiplexers is substantially higher for the **Pipeline Model** compared to the **Concurrent Model**. This is of course the effect that Chapter 7 set out to achieve. Unfortunately the need for wide dynamic range of the **Pipeline Model** would ultimately threaten this gain.

7.5.3.2 Propagation delay

The critical path propagation delays of the **Pipeline Model** core (sweet spot) when implemented on the candidate FPGA devices of Table 7 were calculated with the HDL Coder. These compilations identified as worst-case Critical Path the scenario where a value read from the **Temperature Memory** block and fed into the **Filtered Equation** block in Figure 154 results in a change on the *Next_p* output. For example, it calculated a Critical Path Delay of 104,360 nsec for this path in the case of device 5CEFA9F23C7 from the CYCLONE V family.

This assessment has some problems. The first problem is that Quartus II performs its critical path analysis using typical propagation delays for various blocks implemented in a limited range of device families. The devices that have been characterized for this purpose (thus far) are the CYCLONE V family from Intel, the VIRTEX-7 (speed grade -1) family from Xilinx and the ZYNQ (speed grade -1) family from Xilinx [124]. For a specific device operating at a specific core voltage, case temperature and with unique mapping and routing the propagation delays would therefore be slightly different from those as presented.

Note: *"Data Arrival Time" in Quartus II is equal to Critical Path Delay of the circuit plus a few small terms (e.g. "Clock Uncertainty Time") that amount to about 5 nsec.*

The second problem is created during the Place and Route stage when the the Quartus II compiler allocates specific blocks to specific transistors and tracks within the FPGA. As shown in Figure 172 Quartus II calculated a critical path delay of 304,614 nsec for device 5CEFA9F23C7 from the CYCLONE V family assuming the so-called "Slow 900mV 100C Model" versus the initial estimate of 104,360 nsec. The reason for this big difference is that the Critical Path identified by Quartus II includes the **REC** block inside the **Matrix A Calculation** block. Calculating reciprocals is a computation-intensive task which requires 214,791 nsec in this case. When this fact is taken into account the two estimates are actually quite close.

```
    Info (332115):   303.888     0.000 FF   IC  u_Filtered_Equation|u_Time_Domain_Estimation|Add5~409|cin
    Info (332115):   303.959     0.071 FF CELL  u_Filtered_Equation|u_Time_Domain_Estimation|Add5~409|cout
    Info (332115):   303.959     0.000 FF   IC  u_Filtered_Equation|u_Time_Domain_Estimation|Add5~413|cin
    Info (332115):   304.332     0.373 FR CELL  u_Filtered_Equation|u_Time_Domain_Estimation|Add5~413|sumout
    Info (332115):   304.332     0.000 RR   IC  u_Filtered_Equation|u_Time_Domain_Estimation|Unit_Delay5_out1[103]|d
    Info (332115):   304.614     0.282 RR  CELL
Filtered_Equation:u_Filtered_Equation|Time_Domain_Estimation:u_Time_Domain_Estimation|Unit_Delay5_out1[103]
    Info (332115): Data Required Path:
    Info (332115): Total (ns)  Incr (ns)     Type  Element
    Info (332115): ==========  ========= ==  ====  ================================
    Info (332115):   100.000    100.000          latch edge time
    Info (332115):   101.111      1.111  R       clock network delay
    Info (332115):   101.031     -0.080          clock uncertainty
    Info (332115):   101.031      0.000     uTsu
Filtered_Equation:u_Filtered_Equation|Time_Domain_Estimation:u_Time_Domain_Estimation|Unit_Delay5_out1[103]
    Info (332115): Data Arrival Time  :  304.614
    Info (332115): Data Required Time :  101.031
    Info (332115): Slack              : -203.583 (VIOLATED)
    Info (332115): ===============================================================
```

Figure 177. Part of QUARTUS II timing report for implementing Pipeline Model core (sweet spot) on device 5CEFA9F23C7 [A Landman, 2017]

The third problem with the Critical Paths as identified by both the HDL Coder and Quartus II is that both assumed routes from the *Cur_T* input to the *Nex_T* output of the **Filtered Equation** block as illustrated in Figure 154. In the case of the **Pipeline Model** this is completely wrong since the **Filtered Equation** block is calculated in stages. What is actually relevant here is the longest propagation delay that can occur during any of the "instructions" executed by any of the

seven pseudo-processors of the **Filtered Equation** block. This delay defines the maximum frequency of the *Clk* signal which in turn defines the execution times of the seven pseudo-processors as illustrated in Figure 174.

Inspection of all the pseudo-programs of the **Pipeline Model** shows that the vast majority of pseudo-instructions used compise the same operations as the **Calculate_GG_HH** and **Store_GG_HH** instruction pair as used by the **Inverse A Calculation** pseudo-processor. The operations performed during this instruction together with the propagation delays through associated hardware for Intel device 5CEFA9F23C7 is described in Table 9. Inspection of the table shows that the correspondence between the estimations of HD Coder and Quartus II is not good. One possible reason for the large difference may be the fact that the propagation delays calculated by HDL Coder are based on typical values of the Virtex-7 (speed grade -1) family from Xilinx while the propagation delays calculated by Quartus II are based on Intel device 5CEFA9F23C7 (results achieved after performing the Place and Route process and assuming the "Slow 900 mV 100 C" model). Since Quartus II calculates the propagation delays of a specific Intel device taking into account specific transistor positions and routing, the Quartus values will be used from here onwards

Table 9. Propagation delays of Calculate_GG_HH and Store_GG_HH instructions using device 5CEFA9F23C7 [A Landman, 2017]

No	Operation	Propagation delay (HDL Coder)	Propagation delay (Quartus II)
Clock cycle 1	**SwitchA1** block selects signal *DD* and routes it to input1 of the **Mul1** block. **SwitchB1** block selects signal *A12* and routes it to input2 of the **Mul1** block. Longest propagation delay is caused by **SwitchB1** block.	1,3760 nsec	0,764 nsec
	Mul1 block multiplies the signals generated by the **SwitchA1** and **SwitchB1** blocks.	14,6930 nsec	10,949 nsec
	Inv1 block inverts the output of the **Mul1** block	13,8510 nsec	2,427 nsec
	Total	29,9200 nsec	14,140 nsec
Clock cycle 2	**Switch 1** block selects the signal generated by the **Inv** block and routes it to the **Data Store 6** block.	0,6880 nsec	0,308 nsec
	Data Store 6 block stores the signal generated by the **Inv 1** block.	0,0000 nsec	0,774 nsec
	Total	0,6880 nsec	1,082 nsec

It is evident from Table 9 that the longest propagation delay occurs in clock cycle 1 and that it has a duration of 14,140 nsec in case of Intel device 5CEFA9F23C7. This means that the clock frequency for implementing the **Pipeline Model** on this device may not exceed 70,7 Mhz.

Note: *A few pseudo-instructions use the **Inv1** block which has a propagation delay much longer than the typical value as shown in Table 9. These exceptions to the general pseudo-instruction sequence are rare and has little influence on the overall performance of the **Pipeline Model.** In the actual design they are easily accommodated by adding extra clock cycles to the particular states involved.*

Note: *Propagation delays of different pseudo-instructions will vary because the propagation delays of a given block depends on the word size of the block as well as the mapping and routing of the particular block inside the FPGA. The numbers listed in Table 9 should therefore be considered as average values. Racing problems can be solved by adding additional clock cycles on a case by case basis.*

Finally, we note from Figure 174 that a single minor cycle of the **Pipeline Model** requires 46 clock cycles. In addition, a detailed analysis (not shown here) of the **Pipeline Model** shows that 5 additional minor cycles are added to each major cycle due to the time lag between data sets within the pipeline. Hence, a complete major cycle for a 500x500x500 flight box will require 505 minor cycles or 23230 clock cycles. The minimum time required to perform a major cycle with device 5CEFA9F23C7 will thus be 328 µsec. This translates into a maximum iteration frequency of 3,0 kHz.

Using the same procedure as described above the maximum iteration frequencies of the **Pipeline Model** implemented with the devices as listed in Table 7 were calculated, yielding the results as listed in Table 10. Also shown are the fractions of available ALM, RAM and DSP resources used to implement the **Pipeline Model** core on these devices.

Table 10. Resources used to implement Pipeline Model core (sweet spot) on candidate FPGA devices [A Landman, 2017]

	CYCLONE V	ARRIA 10	STRATIX V	STRATIX 10
Device	5CEFA9F23C7	10AS066H2F34E1SG	5SGSMD8K1F40C2	GX2800
Instruction propagation delay	14,140 nsec	6,438 nsec	9,449 nsec	9,449 nsec (2)
Max clock frequency	70,7 MHz	155,3 MHz	105,8 MHz	105,8 MHz (2)
Max (1) iteration rate	3,0 kHz	6,7 kHz	4,5 kHz	4,5 kHz (2)
ALM use	19349/113560 (17%)	19286/251680 (8%)	18616/262400 (7%)	18616/933120 (1,99%) (2)
RAM use	614400/12492800 (5%)	614400/43642880 (1%)	614400/52572160 (1%)	614400/240721000 (0,26%) (2)
DSP use	111/342 (32%)	111/1687 (7%)	92/1963 (5%)	92/5760 (1,6%) (2)

Note (1) : For a Flight Box containing 500x500x500 cells

Note (2) : Estimated figure (compiler for STRATIX 10 was not available at the time of this study)

The conclusions that can be drawn from Table 10 are as follows:

a) The required iteration rate of 10 kHz cannot be achieved with any of the candidate devices. The iteration rate required to match the Nyquist frequency of the Space/Main

Rotor combination is (only) met with devices from the ARRIA 10, STRATIX V and STRATIX 10 families.

b) The DSP to Fabric Usage Ratio ranges from 1:2 for the CYCLONE V device family to 1:1 for the ARRIA 10 and STRATIX V device families (i.e. an ideal combination for the **Pipeline Model**). The DSP to Fabric Usage Ratio of the STRATIX 10 device family is 2,5:1 which means device GX2800 actually contains too many multipliers for the **Pipeline Model**. A device with less multipliers and more fabric would be a better match.

c) The number of devices required to implement the **Pipeline Model** on a 500x500 Planar Array Processor is determined by DSP use in the case of device 5CEFA9F23C7 of the CYCLONE V family and ALM use in the others. Should device 5CEFA9F23C7 be used it would require a 289x289 array of devices (i.e. 83521 in total), forming a sheet 6,0 x 6,0 meter in size. This would be an order of magnitude more than the total exterior surface area of the M159 Rocket Launcher. Should device GX2800 be used it would require a 45x45 array of devices (i.e. 2000 in total), forming a sheet 2,45 x 2,0 meter in size which would be slightly more than the total exterior surface area of the M159 Rocket Launcher.

7.5.3.3 Some further optimization

It was shown above that implementation of the **Pipeline Model** on a 500x500 Planar Array Processor would exceed the physical constraints of the Precision Rocket Launcher with a small margin. Can the design be improved further to mitigate this problem?

One possibility is to consider some other similar devices of the STRATIX 10 family as listed in Table 11. It is evident from the table that device GX2800 actually yields the most cores per device, even though more than half of the hardware multipliers would be unused.

Table 11. Resources used to implement Pipeline Model core (sweet spot) on selection of STRATIX 10 devices [A Landman, 2017]

	GX2100	GX2500	GX2800	GX4500	GX5500
ALM	679680	821150	933,120	1512820	1867680
RAM	127 Mb	195 Mb	229 Mb	137 Mb	137 Mb
Multipliers	7488	10022	11520	3960	3960
ALM use	2,74%	2,27%	1,99%	1,23%	1,00%
RAM use	0,48%	0,32%	0,27%	0,49%	0,49%
DSP use	1,23%	0,92%	0,80%	2,32%	2,32%
Cores per device	36	44	50	43	43

Another possibility not investigated in detail during this study thus far would be to implement the **Pipeline Model** in single precision floating point notation. In such an implementation each variable would be represented with 32 bits which is even less than the smallest word width (40 bits) of the **Pipeline Model** implemented in fixed point notation at sweet spot. This solution would solve both the resolution and dynamic range problems of the **Pipeline Model** as described in paragraph 7.5.2. This approach was not considered until now because researchers such as *Deest, Yuki, Sentieys & Derrien* [122] as well as *Lin, Talathi & Annapureddy* [123] as

well as *Rosa & Bonato* [125] all assume that a fixed point notation scheme yields the most optimal FPGA solution. In the case of a literal implementation of the Taylored Navier Stokes equation this assumption is true because each hardware multiplier is used for only one calculation, so the position of the decimal point can be optimally chosen for that particular calculation. In the case of the **Pipeline Model** each hardware multiplier is used for several different calculations, so the position of the decimal point cannot be optimized for one calculation only. This results in excessively long word sizes, and it is clear that that the de-facto assumption of the free literature is definitely not the case for the **Pipeline Model**.

To pursue the single precision floating point option one step further a simple Simulink model containing only one single precision multiplier was built, compiled with the HDL Coder and submitted to Quartus II for porting into device 10AS066H2F34E1SG of the ARRIA 10 family. The compilation result showed that only one DSP block was used to implement the single precision multiplier. Combining this fact with the finding of paragraph 7.5.3.1 that the **Pipeline Model** core requires 10 multipliers to implement we conclude that a total of 10 DSP blocks would be required to implement a single **Pipeline Model** core in single precision floating point.

Should the **Pipeline Model** be embedded in single precision notation on the 5SGSMD8K1F40C2 device of the STRATIX 5 family each core will require 10 DSP blocks and about $\left(\frac{18616*10}{92}\right) = 2023$ ALM blocks to implement. In such an implementation a 11x11 array of **Pipeline Model** cores can be packed into each 5SGSMD8K1F40C2 device so that the Planar Array Processor itself will require a 46x46 array of FPGA devices. This will cover a surface area of 1,93 x 1,93 meter or 3,74 m^2 which exceeds the 1,8 m^2 limit of the Precision Rocket Launcher.

Should the **Pipeline Model** be embedded in single precision notation on the GX2800 device of the STRATIX 10 family each core will require 10 DSP blocks and about $\left(\frac{18616*10}{92}\right) = 2023$ ALM blocks to implement. In such an implementation a 21x21 array of **Pipeline Model** cores can be packed into each GX2800 device so that the Planar Array Processor itself will require a 24x24 array of FPGA devices. This will cover a surface area of 1,115 x 1,365 meter or 1,52 m^2 which would fit into the 1,8 m^2 limit of the Precision Rocket Launcher.

7.6 Performance of the Planar Array Processor

Since the Array Processor Modules will all be the same, use of device GX2800 as described above will translate into an overall array size of 525x525 cores. Keeping the grid size at 200 mm such a system will yield a Flight Box of 105x105x105 meter.

Also note that the propagation delays of the **Pipeline Model** as listed in Table 10 are dependant on the fixed point word lengths of the model at sweet spot configuration. Conversion to single precision floating point notation will incur a drastic reduction in word length with concomitant improvement in maximum iteration rate. Determination of this rate should be done by subjecting a single precision floating point implementation of the **Pipeline Model** core to compilation with the HDL Coder and Quartus II. Since a HDL compiler for device GX2800 was not available in Quartus II at the time of performing this study, the maximum iteration rate of such a system was assumed as 10 kHz.

7.6.1 Throughput

To calculate the throughput of the Planar Array Processor with **Pipeline Model** embedded in single precision notation we now proceed to count the total number of operations performed in each stage of the **Pipeline Model** core. The result is illustrated in Table 12 from which we

conclude that the **Pipeline Model** core performs 415 single precision floating point operations/Test Volume/Major Cycle.

Table 12. Floating Point Operations of the Filtered Equation block [A Landman, 2017]

Stage	Operation			
	Mul	Div	Sum	Invert
Spatial Derivatives Calculation	42		21	
Matrix A Calculation	15	2	3	
Matrix B Calculation	16		15	
Matrix C Calculation	43	1	4	11
C x B Calculation	13		8	
Time Domain Estimation	16		10	
Spatial Domain Estimation	10		30	
Totals	310	3	91	11

Hence, the total number of single precision floating point operations O_{mc} performed by the **Pipeline Model** embedded in the Planar Array Processor during one Major Cycle is given by:

$$O_{mc} = 525^3 * 415 \approx 6{,}01.10^{10} \text{ flops/major cycle} \quad \textbf{Equation 121}$$

With the combination running at an iteration rate of 10 kHz, the total throughput O_{tot} is given by:

$$O_{tot} = 6{,}01.10^{10} * 10^4 = 0{,}601 \text{ petaflops} \quad \textbf{Equation 122}$$

Note that the throughput as calculated above does not include the temporary storage of data in buffers (omitted because they are not mathematical operations).

7.6.2 RAM data flow rate

To calculate the total data flow rate to and from RAM of the **Pipeline Model** embedded in the Planar Array Processor it is evident that for each cell, thirty five RAM locations are read and ten RAM locations written during each Major Cycle to cater for the five state variables *p*, *T*, *u*, *v* and *v*. Since the Flight Box will contain 525 x 525 x 525 cells in such a system, the total number of bits R_{mc} exchanged with RAM during one Major Cycle is given by:

$$R_{mc} = 525^3 * 45 * 32 \approx 2{,}084.10^{11} \text{ bits/major cycle} \quad \textbf{Equation 123}$$

With the combination running at an iteration rate of 10 kHz, the total RAM data flow rate R_{tot} is given by:

$$R_{tot} = \frac{2{,}084.10^{11} * 10^4}{8} \approx 0{,}261 \text{ petabytes/sec} \quad \textbf{Equation 124}$$

To put this in perspective, note that this data rate would fill about forty thousand CD-ROM discs in one second.

7.6.3 I/O data flow rate

To calculate the total data flow rate between cores of the **Pipeline Model** embedded in the Planar Array Processor it is evident that each core transmits five current state variables and five temporary state variables to its immediate neighbours during each minor cycle. Since the Planar Array Processor will comprise an array of 525 x 525 cores and each major cycle will (effectively) comprise 525 minor cycles, the total number of bits I_{mc} exchanged between cores during one Major Cycle is given by:

$$I_{mc} = 525^3 * 10 * 32 \approx 4{,}63.10^{10} \text{ bits/major cycle} \qquad \textbf{Equation 125}$$

With the combination running at an iteration rate of 10 kHz, the total I/O data flow rate I_{tot} is given by:

$$I_{tot} = \frac{4{,}63.10^{10} * 10^4}{8} = 0{,}057 \text{ petabytes/sec} \qquad \textbf{Equation 126}$$

7.7 Summary (Develop General Theories)

This chapter was used to perform the seventh Scientific Method Cycle of a family of nested Scientific Method Cycles that yield the concept of a Precision Rocket Launching System. Chapter 2 showed that the downwash of the helicopter must be calculated in real time and with great precision for the PRLS concept to work. To this effect, chapters 0, 5 and 6 were devoted towards developing a suitable algorithm for calculating the downwash flow field and conceiving a Precision Rocket Launcher containing a Planar Array Processor and associated circuit for performing the calculation in real time. That effort showed that if contemporary FPGA devices are used the level of re-use as implied by implementing the **Concurrent Model** on the Planar Array Processor is low enough to achieve the required iteration rate of 10 kHz but too low to ensure that the circuit can fit into the physical limits of the Precision Rocket Launcher. The reason for this problem was found to be the liberal use of hardware multipliers that characterizes the **Concurrent Model**.

The purpose of this chapter was to modify the **Concurrent Model** in such a way as to decrease the use of hardware multipliers to a point where the new circuit can fit within the resources of the Precision Rocket Launcher. Questions were asked as to how this can best be done, leading to the Pipeline Hypothesis which states that it can be achieved by shifting the operational point of the **Concurrent Model** in Techno-Re-Use space, such as to increase the level of re-use at the expense of iteration rate.

The Pipeline Hypothesis was tested by conducting a detailed re-design of the Concurrent Model core. This was done by re-arranging the the **Concurrent Model** core into a systolic (pipeline) configuration comprising seven discrete stages that reflect the form of the Taylored Navier Stokes equation (like multiple waves travelling in a transmission line). Each of these stages performed a functionally seperable part of the original equation that could be calculated for separate moments in time. This allowed for the seven stages to be calculated in parallel with one stage feeding into the next in a pipeline arrangement that was named the **Pipeline Model** for the purposes of this study.

The structure of the **Pipeline Model** incurred a major performance improvement compared with the **Concurrent Model** because the seven stages could run in parallel. This allowed each stage to be implemented with a pseudo-processor (which employs a heavy level of re-use with concomitant performance penalty) that in essense comprised a state machine controlling two switches, a multiplier (the shared element) and a set of data buffers. The resulting mix reduced the number of multipliers used from 157 in the case of the **Concurrent Model** to 10 in the case

of the **Pipeline Model**. The **Pipeline Model** was tested by comparing the movies generated by the Big Box experiment and the **Pipeline Model**. Both models produced the same movie.

Making the same naïve assumption (that fixed point notation yields a superior FPGA design) as most researchers, we scaled the **Pipeline Model** core by hand after the Fixed Point Tool of Simulink proved incapable of performing this function. The conversion was performed by making notation adjustments for each variable (one by one) and running the **Pipeline Model** to compare the result of the change with the original reference data. This process took several weeks, resulting in blocks with word lengths varying between 40 bits and 120 bits (the so-called Sweet Spot configuration). One of the effects noticed during this process was that the resolution requirements of blocks were highest at centre of the pipeline where the inverse of matrix **A** and partial derivatives are calculated. Another effect noticed was that the performance of the **Pipeline Model** is dominated by so-called "cornerstone blocks". As a general rule, all the multiplier and reciprocal blocks resorted under this category.

The HDL files of the scaled **Pipeline Model** core were submitted to the Quartus II compiler for compilation and porting into the same range of FPGA's as listed in Table 7. These experiments showed that the **Pipeline Model** would be able to run at a maximum iteration rate ranging between 3,0 kHz for a low-end device such as the 5CEFA9F23C7 from the Cyclone V device family to 6,7 kHz for a high-end device such as the 10AS066H2F34E1SG from the Arria 10 device family. Device GX2800 of the Stratix 10 family yielded a maximum iteration rate of only 4,5 kHz.

The experiment to convert the **Pipeline Model** into sweet spot fixed point notation showed that none of the devices considered in Table 7 could yield 250000 cores within the physical constraints of the Precision Rocket Launcher under these conditions. The solution to this problem came with the realization that the assumption that fixed point notation yields a superior FPGA design can only be true for designs that employ little on no re-use at all. Such cases can afford to employ narrow word widths because the position of the decimal point of each block can be optimized for the few calculations it performs. In cases of high re-use the word width required becomes very wide in order to comply with the dynamic range requirements of a large number of different calculations, especially in the case of cornerstone blocks. The resource problem of the **Pipeline Model** was overcome by introducing three things in the design:

- The **Pipeline Model** core was implemented in single precision floating point notation which reduced the wordlength of all variables of the system to 32 bits. Apart from solving the dynamic range problem this reduced the number of multipliers used to 10 hardware multipliers/core.

- The reciprocal blocks were implemented with megafunctions that only use registers and boolean logic (no multipliers).

- The Planar Array Processor was populated with the GX2800 device from the Stratix 10 family. Compared with other devices of the same family this device yields the highest number of cores per device.

Finally, the single precision floating point implementation should be able to run at 10 kHz iteration rate because the word widths of most of the cornerstone blocks are reduced from 120 bits to 32 bits.

Note: *The solution as described above could only be evaluated on a theoretical basis because a HDL compiler for the Stratix 10 device family was not available at the time of conducting this study. The conclusions made here should be confirmed at a later stage when such a compiler becomes available.*

The first contribution of this chapter is proof that as at the year 2018 there is in existence at least one point in the re-use space where a bespoke algorithm/hardware combination can calculate with high accuracy the downwash flow field of a helicopter in real time and within the physical limits imposed by an airborne application. As such, it provided proof that one of the fundamental assumptions of the PRLS Hypothesis (that the main rotor downwash can be calculated in real time with equipment on-board a helicopter) is true. It is also evident that apart from the PRLS this bespoke arrangement can be applied to other applications with similar demanding requirements.

The second contribution of this chapter is that as a general rule fixed point notation schemes are not suitable for designs that employ high levels of re-use. Floating point shemes should be used for these systems.

8 Conclusions (Develop general theories for Cycle 1)

"A person who never made a mistake never tried anything new". **Albert Einstein.**

This chapter concludes the first of a family of nested Scientific Method Cycles that investigate the concept of a Precision Rocket Launching System (PRLS) as introduced in Chapter 1. The PRLS Hypothesis was introduced in Chapter 1 and asserts that a Model Predictive Control Loop can serve as a low-cost alternative to Smart Rockets. The PRLS Hypothesis implies a wide range of requirements, one of which is real time estimation of the downwash flow pattern of a helicopter. The design thread created by this requirement was traced by this study, resulting in a range of secondary hypotheses that cross-feed into each other. This chapter is an attempt to consolidate the conclusions reached by these secondary hypotheses in order to make a final judgement as to the validity of the PRLS Hypothesis.

The study as contained in this document originated with the modeling of a helicopter-based rocket launching system. The original brief was open-ended, did not enforce any particular helicopter, rocket or operational scenario and was envisaged to be limited in extent. What transpired was an extensive effort to derive a solution for the rocket accuracy problem that can be implemented on aircraft such as Rooivalk AH-2A.

To complete such a study requires a vast amount of work of which only a sub-set is contained in this document. Completion of the remaining work will require additional tasks to be conducted under other studies as mooted in paragraph 8.19. Nevertheless, some important results, trends and conclusions that are bound to shape the way ahead have been reached. These factors are reflected by a main PRLS hypothesis supported by a corollary of secondary hypotheses as summarized in Table 13.

Table 13. Short summary of hypotheses proven/disproven under this study

Hypothesis	Results and conclusions
PRLS hypothesis	Postulates that a standard Free Flight Rocket such as the MK66 and FZ90 rockets can be delivered with accuracy approaching that of smart rockets through the use of a Model Predictive Control Loop. For this hypothesis to be true the error mechanisms at play must all be understood and modelled with sufficient accuracy and speed so that they can be compensated for in real time. This requirement triggered an investigation that led to a corollary of secondary hypotheses that were all tested as described below. Overall, the results indicate that the PRLS hypothesis is true.
Coning hypothesis	Postulates that the main error mechanism of a Free Flight Rocket launched from a helicopter is perturbation of the rocket trajectory due to weather cocking of the rocket while within the main rotor downwash of the helicopter. This hypothesis was tested by combining an extensive model of the FZ90 Rocket and a rudimentary downwash model of Rooivalk AH-2A. The simulation results showed strong correspondence with observational data. We concluded that the Coning Hypothesis is true and that the downwash needs to be predicted in real time and with sufficient accuracy to identify vortices with spatial frequencies as high as 5 cycles/meter and temporal frequencies as high as 10 kHz.
Flexible Disk hypothesis	Postulates that the flow through the main rotor of a helicopter can be represented as a 3-dimensional surface of micro-jets (current

	controlled current sources) that morphs into different shapes as function of things such as main rotor rotation rate, cyclic and collective lever positions (i.e. the position and orientation of the swash plate) and the velocity distribution of air flowing into the surface. Flows generated by the blades are then modelled by enabling those micro-jets direcly below the blades with other micro-jets disabled. Assuming constant inflow velocities over the disk this technique yields outflow velocities that are close to those observed in reality. This serves as strong indicator that the accuracy of the model is sufficient for PRLS purposes. Ultimately the model must be proven by combining it with the downwash model as described below. This technique will yield the detail in the estimated downwash flow field as called for through the Coning Hypothesis.
Donut hypothesis	Postulates that the downwash flow field of a helicopter is governed by the Navier Stokes equations with boundary conditions dictated by various (non-stationary) bodies such as the main rotor blades, fuselage and ground. In addition, the Navier Stokes equations can be written in a form conducive for implementation on a dedicated computer, such that the resulting equation can be solved at an iteration rate in excess of 10 kHz. This hypothesis was tested by deriving the Navier Stokes equations from first principles and writing it in a matrix notation referred to as the Taylored Navier Stokes Equation. This equation is amenable to optimization (sparse matrix) and to parallel processing because it can be packaged into a standalone cell. Thousands of these cells can run in parallel to calculate a downwash flow field solution. A functionally equivalent system was implemented in Matlab and stimulated with an outflow as calculated with the Flexible Disk hypothesis assuming constant inflow. This first-order simulation yielded results close to those observed in reality and showed that the computational volume of a helicopter is as large as a rugby field.
Automata hypothesis	Postulates that it should be possible to invent a 3-dimensional array of cells (Array Processor) with each cell containing a copy of the Taylored Navier Stokes Equation in a circuit comprised of a number of interconnected FPGA devices. It should be possible to arrange this circuit in such a way that it acts like a network of cellular automata with each cell performing exactly the same function in lockstep with each other as clocked by a free-running state machine. Employing dual-port memory it should be possible to construct a **Commutation** process that concurrently manipulates the internal states of individual cells, such that the excitation and deflection effects of objects such as a main rotor blade, fuselage, ground and gas flows from engine exhausts can be factored into the flow field solution. It should also be possible to construct a **Probing** process that concurrently probes the solution as calculated by the array and a **Grid Hop** process that allows the system to analyze adjacent volumes of space. Finally, it should be possible to embed such a circuit within a Precision Rocket Launcher similar in size to the M159 Rocket Launcher as used on Rooivalk AH-2A. This circuit should have a throughput suitable for implementing and running the **Gas Flow**, **Commutation**, **Probing** and **Grid Hop** processes at 10 kHz

	iteration rate. To prove this hypothesis a design of a Precision Rocket Launcher was performed, illustrating the fact that a 25x25 array of top-end FPGA devices can be installed on the outer periphery of a launcher containing nineteen FZ90 Rockets. This circuit has a maximum throughput of 5,75 petaflop which is well above the 0,601 petaflop required. Power dissipation of the circuit is 9,1 kW. This preliminary design proved that the Automata hypothesis is true.
Concurrent hypothesis	Postulates that it should be possible to: ▪ Design a bespoke Array Processor circuit compatable with the resources of the Precision Rocket Launcher. ▪ Design this circuit such that it comprises hundreds of thousands of cells running concurrently with each cell containing a copy of the Taylored Navier Stokes Equation and its solver. ▪ Coordinate all of these cells with a central state machine that performs a known and repeating state transition sequence that avoids the housekeeping limit of Amdahl. ▪ Implement this circuit in fixed point notation and run it at an iteration rate in excess of 10 kHz. ▪ Design and test the circuit using Hardware Development Language within the Simulink Integrated Development Environment. The Concurrent hypothesis was tested by performing a detailed design of the Array Processor using the Fixed Point Tool and HDL Coder in conjunction with Quartus II within the Simulink design environment. The resulting design did not fit within the resources of the Precision Rocket Launcher, proving the Concurrent hypothesis false. This result showed that a higher level of re-use is required than that implied by the Concurrent hypothesis.
Pipeline hypothesis	Postulates that it should be possible to: ▪ Design a bespoke circuit that implements each cell of the Downwash Model as a one-dimensional systolic processor (pipeline), causing an increase in the level of re-use as implied by the basic Array Processor. This increase in re-use should enable reduction of hardware multipliers required, such that the overall circuit will fit within the resources of the Precision Rocket Launcher. ▪ Coordinate all of these cells with a central state machine that performs a known and repeating state transition sequence that avoids the housekeeping limit of Amdahl. ▪ Implement this circuit in fixed point notation and run it at an iteration rate in excess of 10 kHz. The Pipeline hypothesis was tested by performing a detailed design of the Array Processor using the Fixed Point Tool and HDL Coder in conjunction with Quartus II within the Simulink design environment. The resulting design did not exceed the resources of the Precision Rocket Launcher, proving the Pipeline hypothesis true.

8.1 The PRLS is viable

It was shown that the mission system of combat helicopters such as Rooivalk AH-2A can be modified in such a way that the rocket accuracy problem can be solved without incurring the repeating cost penalty of smart rockets. This was one of the two system needs that ultimately drove this entire study and led to formulation of the PRLS Hypothesis and its corollaries.

A wide range of models were built and various simulations run to prove the viability of many aspects of the PRLS Hypothesis ranging all the way from system level right down to component level. This was done within the framework of a number of Scientific Method Loops that fed into each other, calling for a research process with iterative nature.

Nothing that would invalidate or prevent the implementation of the PRLS concept was found. In contrast, some of the results achieved support the PRLS concept:

- The dynamics of the FZ90 Rocket could be modelled.
- The trajectory of the FZ90 Rocket could be predicted with great precision.
- The downwash and its interaction with the ground could be calculated and the behavior of the rocket within the downwash could be calculated.
- Algorithms for modelling the flow generated by the main rotor and the response of space to this stimulus could be formulated.
- It was shown that the Taylored Navier Stokes Equation can be solved in real time using an array of FPGA devices.
- The dynamic inter-actions between the sub-systems as listed above could be calculated.

The joint conclusion from these successes is simple – the response of any given Rocket Launching System can be calculated in real time for any given set of conditions, so the rocket can be aimed with a Model Predictive Control Loop. This will improve the accuracy of weapon systems such as the Rooivalk AH-2A Rocket Launching System with two orders of magnitude.

8.2 The rocket accuracy problem is universal

It was shown that the coning problem is not unique to Rooivalk AH-2A. It is also present on other helicopters so that the solution as proposed by this thesis can be applied in other helicopters with particular adaptations as may be required. From a business point of view this means that there is a gap in the weapon system market between the poor performance of (low cost) standard Wrap Around Fin Rockets and the highly accurate performance of (expensive) smart rockets.

8.3 Downwash can be calculated in real time

It was shown that it is possible to estimate the downwash of a helicopter in real time by implementing the Taylored Navier Stokes equation in an array of Field Programmable Gate Arrays. Two designs were generated. The Concurrent Model can estimate the downwash using a computational domain comprising 500x500x500 cells at an iteration rate of 63,4 kHz when implemented on Intel device GX2800 of the Stratix 10 family. The Pipeline Model can estimate the downwash using a computational domain comprising 525x525x525 cells at an iteration rate of 4,5 kHz when implemented on Intel device GX2800 of the Stratix 10 family. The iteration rate

of the Pipeline Model can probably be increased to close to 10 kHz if the model is converted into single precision floating point notation.

8.4 Amdahl's law can be overcome

It was shown that the housekeeping function which forms the backbone of Amdahl's law can be completely eliminated by implementing the system being modelled as an array of cellular automata running in lockstep with a central state machine that executes a fixed operational cycle. Such a system is ideally suited for implementation with an Array Processor comprising an array of Field Programmable Arrays which avoids the Processor Bottleneck, Memory Bottleneck and I/O Bottleneck that plague Von Neumann processors.

Depending on the geometric configuration chosen, expansion (speedup) of such an Array Processor is only limited by the worst-case propagation delay (typical about 100 nsec) within the circuit. In the case of the PRLS this caused the performance of the Array Processor to be directly proportional to the number of Test Volumes in the Computational Domain, yielding a behavior which is in agreement with Gustafson's law except that the Array Processor does not spend any time on any sequential task.

8.5 A hybrid processor

It was shown that the performance required to calculate the downwash of a helicopter in real time can be obtained by constructing a processor comprising an array of cellular automata with each cell performing exactly the same function (Taylored Navier Stokes equation in this case) in lockstep with each other as clocked by a central state machine that executes a fixed operational cycle.

This yields a dedicated processor that is very powerful but not easily modified. The resulting Downwash Model can then be combined with other models running on Von Niewman machines which are easily modified, yielding the best of both worlds.

8.6 Performing only one task

It was shown that the Array Processor as described in this thesis is very fast, but that it can do only one thing. To design and build such a processor by hand is a very labour intensive undertaking which we have tried to avoid by using standard building blocks in an Integrated Development Environment (Simulink in conjunction with the Fixed Point Tool and HDL Coder).

The quest to harness the immense power of an FPGA by re-configuring the device at run time has been persued by *Van der Bauwhede et al* [112] of Glasgow University for several years. While such an objective may well be possible, our experience shows that it is a tedious and labour-intensive process.

8.7 Handling big-size problems in Simulink

It was shown that when running long simulations or constructing big Simulink models the limitations of Simulink and the Fixed Point Tool can be overcome through the principle of "Equivalence of State". After that hardware in the loop principles are required to ensure the design process can be performed with short turnaround time.

This study has shown that the Fixed Point Tool of Simulink is intended for relatively small systems compared with the Array Processor as developed in this thesis. Some models took several days to run on the Fixed Point Tool and often caused the program to crash.

8.8 A mesh of 145 million test volumes

It was shown that it is possible to fit and cool an Array Processor that models a flight box comprising 145 million cells (control volumes) within the confines of the Precision Rocket Launcher. This will be enough to estimate the downwash of a helicopter with enough accuracy to make the PRLS viable.

8.9 Commutation & Probing is viable

It was shown that the concepts of Commutation and Probing are viable. To do this suitable test-of-concept algorithms were conceived and associated circuits developed. The mechanisms were demonstrated to be workable albeit computation intensive. The refresh rate (inverse of time taken to transfer a complete shape matrix into the array) must be at least 5,5 kHz to comply with the Nyquist criteria (invoked by Rotor Blades cutting a given cell of the array). Choosing a safety factor of 2 sets the required iteration rate at 10 Khz.

To achieve such a rate the commutation process will have to be shared between a Commutation Processor located in the Main Electronics Unit and Local Commutation Processors located on each Array Processor Module.

8.10 Two solutions for the Array Processor

It was shown that there are in existence at least two solutions for an Array Processor capable of predicting the main rotor downwash of a helicopter in real time.

The **Concurrent Model** is relatively easy to implement and can solve the Taylored Navier Stokes equation at a rate as high as 63,4 kHz using contemporary FPGA devices. However, the **Concurrent Model** is multiplier intensive which results in an unfavourable Multiplier to Fabric Ratio, typically 10:1 for top-end devices such as the STRATIX 10 device family. In practice this would mean a large quantity of FPGA resources that cannot be used, unnecessary heat dissipation, unnecessary real estate usage, more mass and escalated price of construction.

The **Pipeline Model** is more difficult to design and can currently solve the Taylored Navier Stokes equation at a rate as high as 5,5 kHz using contemporary FPGA devices. This rate can be increased to 10 kHz by implementing the design in single precision floating point notation. The **Pipeline Model** yields a design which has a favourable Multiplier to Fabric Ratio, typically 4:5.

8.11 Standard building blocks

Various Simulink blocks such as the **FZ90 Rocket Model**, **Main Rotor Model**, **Downwash Model** and **Pipeline Model** were developed under this study. These blocks form discrete entities in the overall Data Flow Block diagram of an aircraft such as Rooivalk AH-2A. As such they represent standard building blocks which can also be used in other applications. They all use relatively simple interfaces with other models of elements such as Main Rotor, Structure,

Engines and Precision Rocket Launchers which makes it possible to re-configure the overall model for other applications.

Combined with the Array Processor the **Downwash Model** can be used to determine the air flow pattern over any combination of objects or within a containing vessel such as a wind tunnel. In the case in a chemical reactor the model can be used as part a Model Predictive Control Loop to control a high-speed chemical reaction.

8.12 The Flight Box is as big as a rugby field

It was shown that a reasonable size for the computational domain of an 8-ton helicopter like Rooivalk AH-2A is a 100x100x100 meter cube. The wind velocity at the outer edges of such a computational domain is less than 1m/s for such a helicopter.

8.13 The Taylored Navier Stokes equation

A tailored version of the Navier Stokes Equation was developed for the purposes of this study, including a stable solver similar to a real time solver as developed by Stam [67] [68] [69].

8.14 Commutation is like graphics

The Commutation process is based on manipulating objects which are constructed with triangles. This is similar to the type of processing performed by Graphic Processor Units in computer games. Due to the relative ease with which a GPU can be adapted using the CUDA language, the Commutation Process is best implemented with a Graphics Processor Unit.

8.15 A new way to develop software

All the software produced under this study (with the exception of the **Variable Mass Body** block) were designed and compiled using Data Flow Block diagrams in the Simulink Integrated Development Environment. No conventional software written in a High Level Design Language such as C or HDL was designed by hand.

8.16 The PRL is a supercomputer

A conceptual design of a 19-tube Precision Rocket Launcher was generated. The dimensions of the design was found to be comparable with the dimensions of the current M159 Rocket Launcher as used on Rooivalk AH-2A. The design was shown to have internal layout suitable for housing and cooling a 25x25 array of Intel GX2800 devices which in turn can be programmed to contain a 525x525 array of cores implementing the Taylored Navier Stokes equation. Absolute maximum throughput of such an Array Processor is 5,75 Petaflop.

8.17 ASIC is not required

It was shown that the Concurrent Model yields a design with Multiplier to Fabric Ratio close to unity. Hence, use of an Application Specific Integrated Circuit (very expensive to design, modify and fabricate) is not required for implementing the Array Processor.

8.18 A model for general use

It was shown that the computational segment of the PRLS can be installed on Rooivalk AH-2A. As such, it is also suitable for installation on other helicopters. This characteristic is due to the following:

- The PRLS Model is based on Simulink and uses standard blocks from a number of Simulink toolboxes [126].
- A number of standard building blocks were developed for use in the PRLS. These blocks are well-documented for study. Some are fitted with Graphic User Interfaces that make them easy to re-configure.
- The original PRLS data flow block diagram can be easily modified and blocks added and removed as required. New dedicated blocks can be developed using S-functions.
- The PRLS model was constructed keeping the Model Based Design paradigm in mind. As a result, the PRLS model represents an Executable Specification that can evolve with time. It also allows for integration with requirements management tools such as DOORS [131] and CORE [132].
- Embedded code destined to run in target hardware can be produced directly from the PRLS model and complies with DO178C [20] provided the correct SIMULINK [126] tools are used. Target hardware can be integrated with the PRLS model using the hardware-in-the-loop facility of the Real Time toolbox.

8.19 Recommendations

In some cases the research as performed under this study provided only partial answers or created new questions. Possible studies and areas of research to answer these questions are listed below.

8.19.1 Explanation for the torque curve of the FZ90 Rocket

A first-order explanation for the shape of the torque curve of the FZ90 Rocket is given in paragraph 2.5.6.3. This explanation needs further investigation and validation. An analysis of the flow within the nozzle of the MK66 Rocket has been done by *Sribonfha, Sirisup & Chusilp* [127] but the behaviour as observed remains an enigma. Work on this topic has already been initiated by a student of the University of the Witwatersrand (Johannesburg, South Africa) with Dr. Randall Paton serving as mentor.

8.19.2 Main Rotor Model expanded

The **Main Rotor Model** should be taken through further increments in sophistication as described in paragraph 3.6 to model the wakes produced by individual blades.

8.19.3 Main Rotor and Space models combined

Even though the **Main Rotor Model** and **Downwash Model** were developed under this study time limitations did not allow for simulation of the helicopter downwash flow pattern with these models combined. Flow patterns under various flight conditions should be evaluated when performing these additional simulations.

8.19.4 Construction of the Array Processor

A scaled-down version of the Array Processor as described in Chapter 5 should be constructed and a mechanism to program multiple FPGAs in-situ with large numbers of Cores performing the same algorithm should be investigated. The circuit should be capable of performing the Grid Hop process as described in paragraph 5.5.4.

The resulting hardware should be integrated with a host computer (laptop) for downloading of the algorithm after compilation with the HDL coder. A mechanism for the real-time recording of minimum and maximum values of all variables of the algorithm running on the Array Processor should be designed and demonstrated. Finally, the Array Processor should be integrated with a suitable Simulink model and evaluated using Hardware-in-the-Loop techniques within the Simulink design environment.

8.19.5 Construction of the Commutate & Probe Processor

An investigation as to how the Commutation and Probing Processes as described in Chapter 5 can be implemented in contemporary electronics is required. The investigation should include a detailed circuit design and validation of the resulting circuit within the Simulink environment.

The design should be able to perform the intersection process as described in Chapter 5, processing 10000 Shape Matrixes per second with each Shape Matrix containing 50000 triangles.

8.19.6 Construction of a Thermal Contol System

A thermal equivalent of the Precision Rocket Launcher and Array Processor Modules as described in paragraph 5.5.9 should be constructed and evaluated.

8.19.7 The PRLS on a real aircraft

Ultimately, the PRLS would have to be implemented on a real aircraft for testing the PRLS Hypothesis in real life. A possible approach (case study) to implement the PRLS on an aircraft such as Rooivalk AH-2A is presented below. Such an undertaking will not be cheap, but if it is done it will generate a range of sub-hypothesis that in most cases would be in need of investigation.

A body of work not described in this study has led to identification of 21 of the main error sources of a helicopter-based rocket launching system. These error sources are listed in Table 14 together with their proposed solutions (sub-hypothesis). Each of the solutions as listed would

have to be implemented for the PRLS Hypothesis to prove true. Note that the assessment of the Rocket Accuracy Problem as contained in Table 14 was the origin of this study. Inspection of Table 14 reveals that only a sub-set of the problems as indicated were investigated in this study - the rest is still to be done.

Table 14. Embodiment of the PRLS system on Rooivalk AH-2A [A Landman, 2014]

Item	Problem	Solution (Sub-hypotheses)
1	Weather cocking and precession of the rocket	Upgrade the Omni-directional Air data Sensor and introduce Precision Rocket Launchers and Two-Axis Precision Cradles at the four inner weapon stations. Use these components to model the rotor downwash and predict the weather cocking and precession of rockets to be launched from each weapon station in real time. Use the results to compensate for these effects by making precise large-signal aiming corrections around azimuth and elevation of each weapon station prior to launch.
2	Twisting Stub Wing	Introduce rate of rotation sensors in the Base Plate of each Two-Axis Precision Cradle, and control these Base Plates around their azimuth and elevation axes such that rotations around these axes at frequencies above 5 Hz are driven towards zero. Model the Stub Wings and weapon stations to estimate the resulting twist of the Stub Wings due to previous rocket launch events. Use the results to compensate for this problem as described under (1).
3	Boresight divergence	Boresight all weapon stations along the aircraft centre-line and model the inter-action between pending rockets and rotor downwash at each weapon station in real time. Use the results to compensate for this problem as described under (1).
4	Variation in rocket geometric characteristics	Design the Precision Rocket Launchers to actively measure the rocket type in each launch tube. Use this information to manage the selection of rockets for rocket cocktails and to predict the trajectory of each pending rocket. Use the results to compensate for this problem as described under (1).
5	Tail-tipping during launch	Model the inter-actions between each pending rocket and its associated Precision Rocket Launcher, Stub Wing and Precision Cradle. Use the results to compensate for this problem as described under (1).
6	Change in super elevation due to ambient wind	Introduce a suitable downwash model in the OAS sensor to produce a better estimate of the ambient wind velocity. Use this information to model the inter-action between each pending rocket and ambient wind. Use the results to compensate for this problem as described under (1).
7	Downwash IGE versus downwash OGE	Use the height above ground information as produced by the Radio Altimeter to manipulate a boundary condition in each Rotor Downwash Model that resembles the ground.
8	Non-homogenic downwash	Introduce a position sensor on the Main Rotor and send this information to each Precision Rocket Launcher.

Item	Problem	Solution (Sub-hypotheses)
		Use this information to manipulate a set of boundary conditions in the Downwash Model to represent the flow caused by Main Rotor and Tail Rotor. Use the resulting flow pattern to predict the trajectory of each pending rocket and schedule launching of each rocket such as to minimize interference between Rockets and the wakes produced by the Main Rotor Blades.
9	Ambient wind turbulence	Inter-action between a flying roket and turbulence in the ambient air is relatively brief because the FZ90 Rocket flies at more than Mach 3. Accept this error as part of the rockets error budget.
10	Gravity drop	Predict the trajectory of each pending Rocket in real time taking into account the effect of gravity. Use the results to compensate for this problem as described under (1).
11	Drag	Predict the trajectory of each pending Rocket in real time taking into account the effect of drag. Use the results to compensate for this problem as described under (1).
12	Variation in motor thrust with temperature	Design the Precision Rocket Launchers such as to actively measure the bulk temperature of each pending rocket. Use this information to predict the trajectory of each pending rocket and use the results to compensate for this problem as described under (1). Also derive suitable partial derivatives to linearize the latest trajectory prediction and make small aiming adjustments as the rocket moves through its launch tube during launch. Introduce levitation coils or micro air jets as described by *Sahoo, Annaswamy & Alvi* [129] in each launch tube to provide the required steering at high enough speed.
13	Variation in motor thrust with batch	Design the Precision Rocket Launchers to measure the actual thrust produced by each pending rocket during launch and compensate for it by releasing the rocket at a specific point in its thrust generation sequence and by steering the rocket as it moves through its launch tube as in (12) above.
14	Thrust mis-alignment	Measure the thrust mis-alignment of the pending rocket during launch and compensate for it as the rocket moves through the launch tube as in (12) above.
15	Low skimming angle	Design a Programmable Time Fuze for installation on each FZ90 Rocket instead of the current Impact Fuze. Estimate the flight time of the pending rocket to the Target while predicting the trajectory of the pending rocket. Design the Precision Rocket Launchers to program, during launch, the Time Fuze of the pending rocket with the estimated flight time to ensure detonation of the Warhead at the target.
16	HMSD error	Modify the Helicopter Mission Computers and Weapon Computers to introduce a Continiously Calculated Release Point (CCRP) weapons mode where the human operator (i.e. Pilot) is removed from the aiming loop while the PRLS performs the aiming and launching

Item	Problem	Solution (Sub-hypotheses)
		of rockets. Use the Main Sight, INS or a remote sighting device to acquire the position of the target with high precision. Note that in such an arrangement the designation of targets and initiation of rocket deliveries will become mainly the task of the WSO and not the Pilot as is currently the case (i.e. an offensive role). Display suitable flight ques (e.g. a tunnel in the sky) on the Pilot HMSD to aid the pilot in keeping the aircraft within a flight envelope suitable for precision rocket delivery.
17	Timing error	Design the Two-Axis Cradles and Precision Rocket Launchers such that they talk to each other directly via the MIL-STD-1760A Aircraft Store Interface (ASI).
18	Coriolis force	Perform all ballistic calculations in Inertial Axes and transform the result into Local Earth Axes or Aircraft Axes as required.
19	Pilot error	Modify the Sight Grips, Sight Control Panel, Weapon Computers and Helicopter Mission Computers such that the WSO is able to manage and select rocket as required and steer the Two-Axis Precision Cradles until the Pipper is located on the target. Use the process as described in (1) to determine the position of the Pipper.
20	OAS error	Install two new air data sensors based on Light Detection and Ranging (LIDAR) technology that can sense wind velocity in front of each Stub Wing to an accuracy of 0,5 ±0,25 m/s. Hybridize the outputs of these sensors with the downwash models to remove the noise in these signals caused by the chaotic flows in the rotor downwash.
21	Main Sight error	Ensure the Main Sight is properly calibrated and bore sigted at all times and accept residual error as part of PRLS error budget.
22	INS error	Replace current navigation sensors with two high-precision Inertial Navigation Systems that are hybridized with Global Positioning System (GPS) receivers.

The envisaged architecture block diagram of the new Mission System of Rooivalk AH-2A with PRLS embedded is illustrated in Figure 178. Apart from modifications to the Helicopter Mission Computers (HMC), Weapons Computers (WEAC) and Helmet Sight Electronic Units (HSEU) the basic design approach of the system – that of an Integrated Management System (IMS) controlling a number of intelligent Front Ends [128] remains the same. The main changes to the current Mission System of Rooivalk AH-2A will be as follows:

a) Replacement of the Omni Directional Airspeed Sensor (located at top of Main Rotor) with two Any Direction Air data Sensors (ADAS) located in the leading edge of each Stub Wing. These devices will employ the Light Detection And Ranging (LIDAR) principle to measure wind velocity in front of the weapon stations with an accuracy of 0,5 ± 0,25 m/s as described by *Bogue & Jentink* [133].

b) Replacement of the SAL29 Actuator and Cradle as currently used on the Inner Weapon Sations of Rooivalk AH-2A with Two-Axis Cradles (TAC).

c) Replacement of the M159 Rocket Launcher and REU as currently used on the Inner Weapon Sations of Rooivalk AH-2A with Precision Rocket Launchers (PRL).

d) Introduction of two Main Rotor Position Sensors (MRPS).

The physical layout of the PRLS on Rooivalk AH-2A will essentially be the same as the current layout of the Aircraft. The locations of devices such as the Weapon Stations, Cannon and Main Sight are evident from Figure 178. Note that the Pilot is in the rear cockpit and the WSO in the front. The ADAS sensors will not change the exterior shape of the Aircraft as they will be mounted on the inside of the leading edge fairing of each Stub Wing.

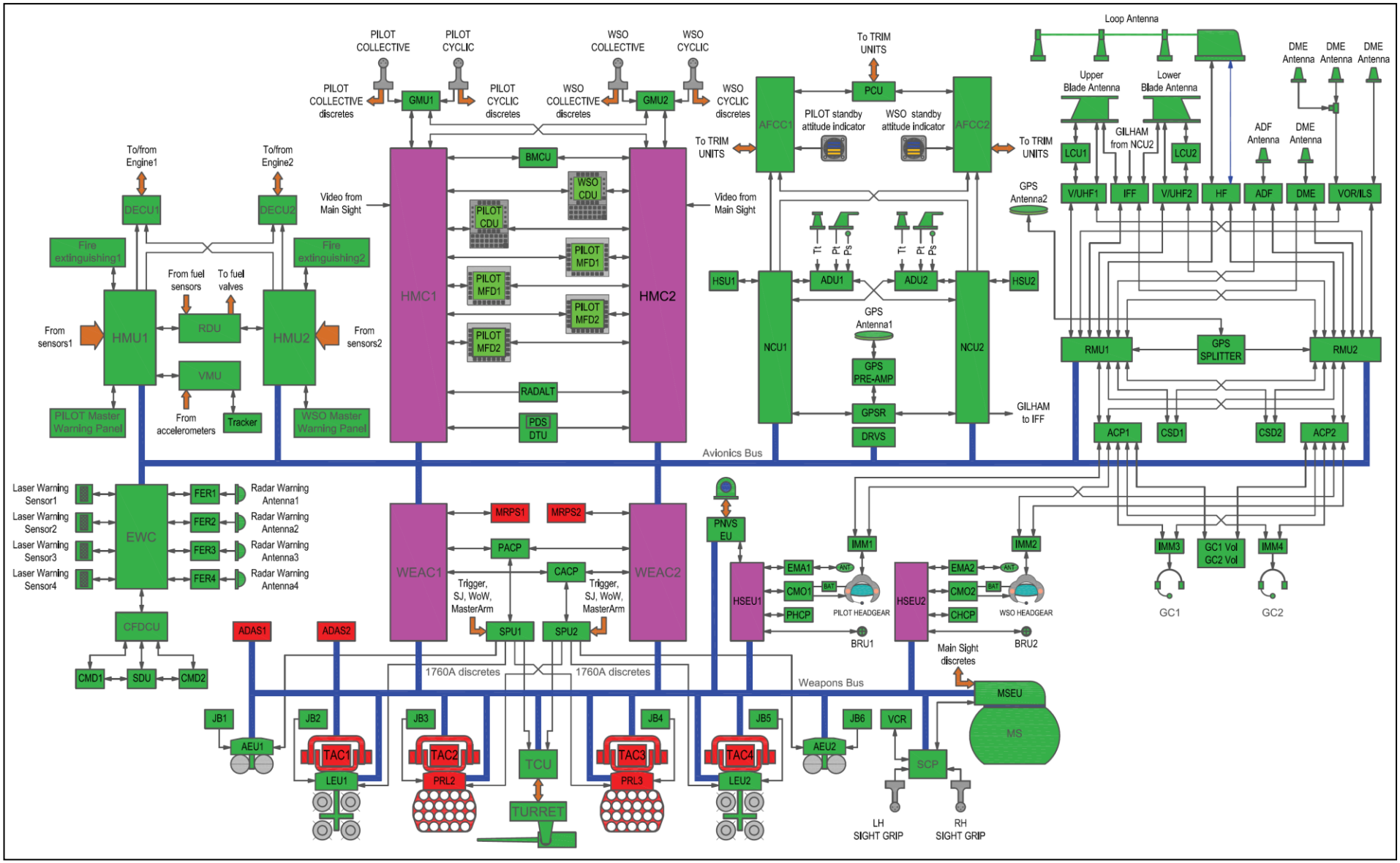


Figure 178. Architecture block diagram of proposed Rooivalk AH-2A Mission System [A Landman, 2014]

References

1 Martin Windrow. “The Algerian War 1954-62”. Osprey Publishing, 1997.

2 Mao Zedong. “The Red Book of Guerrilla Warfare”. El Paso Norte Press. 1937. ISBN 9781934255278.

3 Jonathan Bernstein. “US Army AH-1 Cobra Units in Vietnam“. Osprey Publishing, 2003.

4 Chris Bishop. “Apache AH-64 Boeing (McDonnell Douglas) 1976–2005“.

5 Jean-Denis G.G. Lepage. “Aircraft of the Luftwaffe, 1935–1945: An Illustrated Guide”. McFarland & Company, Inc., Publishers, 2009.

6 Robert M. Cassidy. “Renaissance of the Attack Helicopter in the Close Fight”. ResearchGate, 06 May 2016.

7 Andrew S. Groenke. “CAS, INTERDICTION, AND ATTACK HELICOPTERS”. NAVAL POSTGRADUATE SCHOOL, MONTEREY, CALIFORNIA. June 2005.

8 Benjamin S Lambeth. “Air Power against Terror”. RAND Corporation, National Defence Research Institute. ISBN 0-8330-3724-2. 2005.

9 Mark W. Wonnacott. “Modeling in the design and analysis of a hit-to-kill rocket guidance kit”. Monterey, California. Naval Postgraduate School, 1997-09.

10 Stephen S. Kim and Stacy Walbridge. “High Torque Nozzle Development for the 2.75-Inch Rocket System”. JANNAF Rocket Nozzle Technology Subcommittee Meeting [21st], Cocoa Beach, Florida, March 27-30 2001.

11 Eric Hawley. “MK66 ROCKET MOTOR/HELICOPTER COMPATIBILITY PROGRAM”. 27 March 2003”. https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2003/gun/haw.pdf. Accessed 26 November 2017.

12 Eric Hawley. “Advanced Propulsion Concepts for the HYDRA-70 Rocket System”. 27 March 2003. https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2003/gun/hawl.pdf. Accessed 09 December 2017.

13 John A. Nagl, “Learning to Eat Soup with a Knife. Counterinsurgency Lessons from Malaya and Vietnam”, ISBN 9780226567709, University of Chicago Press, September 2005.

14 Helmoed-Römer Heitman, “Countering Special Forces”, 1 Military Printing Regiment, South African Army Journal, Issue 1 of 2007, pages 9 – 16.

15 Daryl G. Press, “URBAN WARFARE: OPTIONS, PROBLEMS AND THE FUTURE”, MIT Security Studies Program, Hanscom Air Force Base, Bedford, Massachusetts, 20 May 1998.

16 Neta C. Crawford, “US Budgetary Costs of Wars through 2016: $4.79 Trillion and Counting Summary of Costs of the US Wars in Iraq, Syria, Afghanistan and Pakistan and Homeland Security”, Boston University, COST OF WAR, September 2016.

17 The Economist. “Rockets galore”. September 29th 2012, http://www.economist.com/node/21563702, Accessed 21 November 2017.

18 Rory Tingle. “Canadian special forces sniper kills an ISIS fighter from two miles away in the longest confirmed kill shot in history”. MAILONLINE, http://www.dailymail.co.uk/news/article-4628224/Canadian-sniper-kills-ISIS-fighter-TWO-MILES-away.html, Accessed 07 October 2017.

19 Harry Nyquist. "Certain topics in telegraph transmission theory". Trans. AIEE. 47: 617–644 April 1928.

20 Radio Technical Commission for Aeronautics, “Software Considerations in Airborne Systems and Equipment Certification”, DO-178C. 2012.

21 C. Wayne Dahlke and George Batiuk. “HYDRA 70 MK66 AERODYNAMICS AND ROLL ANALYSIS”. US ARMY MISSILE COMMAND, Redstone Arsenal, Alabama. July 1990.

22 BERNER,C., ABATE,G., DUPUIS,A. “Aerodynamics of Wrap Around Fins using Experimental and Computational Techniques”. RTO AVT Symposium on Missile Aerodynamics, Sorrento, Italy, 1998.

23 William H. Mermagen and Vural Oskay. “YAWSONDE TESTS OF 2.75-INCH Mk-66 MOD1 ROCKET”. US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland. August 1981.

24 Peter Daniels and Samual R. Hardy. “Theoretical and Experimental Methods in the solution of Missile Nonlinear Roll Problems”. NAVAL SURFACE WEAPONS CENTER, January 1978.

25 Edward N. Lorenz. “Deterministic Nonperiodic Flow”. Journal of the Atmospheric Sciences, 1963.

26 Geoffrey Ingram Taylor. "Stability of a Viscous Liquid contained between Two Rotating Cylinders". Philosophical Transactions of the Royal Society, January 1923.

27 Harry L. Swinney and Jerry P. Gollub. “The transition to turbulence”. Physics Today 31.8 (1978).

28 Bum Seok Lee, Jae Hoon Choi and Oh Joon Kwon. Numerical Simulation of Free-Flight Rockets Air-Launched from a Helicopter. JOURNAL OF AIRCRAFT, Vol 48, No 5, September-October 2011.

29 American DoD. “Military Handbook DESIGN OF AERODYNAMICALLY STABILIZED FREE ROCKETS”. MIL-HDBK-762(MI) 17 July 1990.

30 Harold J Breaux, “A CRITIQUE OF THE BELL HELICOPTER TEXTRON COBRA 3.75 INCH ROCKET BALLISTIC ALGORITHM”, ARBRL-TR-02463, US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND, Ballistic research laboratory, January 1983.

31 Dean C. Karnopp, Donald L. Margolis and Ronald C. Rosenberg. “System Dynamics. Modeling. Simulation and Control of Mechatronic Systems”. Wiley. ISBN: 978-0-470-88908-4. 2012.

32 R. C. Brown et al. “Six-Degree-of-Freedom Flight Path Study Generalized Computer Program”. AFFDL Report FDL-TDR-64-1, US Air Force Flight Dynamics Laboratory, Wright-Patterson AFB, OH, October 1964.

33 Isidor Kokalari, Theodor Karaja and Maria Guerrisi. “Review on lumped parameter methods for modelling the blood flow in systemic arteries”. J. Biomedical Science and Engineering, January 2013.

34 Herbert H. Woodson and James R. Melcher. “Electromechanical Dynamics”. John Wiley & Sons, Inc.

35 James R. Wertz and Wiley J. Larson. “SPACE MISSION ANALYSIS AND DESIGN Third Edition’. Space Technology Library, 1999.

36 Paulo Flores. “Concepts and Formulations for Spatial Multibody Dynamics”. SpringerBriefs in Applied Sciences and Technology, 2015, ISBN 978-3-319-16189-1.

37 Bruce A. Campbell and Samuel Walter McCandless, Jr. “Introduction to Space Sciences and Spacecraft Applications”. Gulf Publishing Company, 1996.

38 Murray R Spiegel. “Theoretical Mechanics with an introduction to LAGRANGE’S EQUATIONS and HAMILTONIAN THEORY”. McGraw-Hill International Book Company, New York. February 1967.

39 J. M. Coulson and J. F. Richardson. “Chemical Engineering”. Pergamon Press, 1970.

40 NAR Standards and Testing. “ROCKETVISION F32”. November 2000.

41 P. Šaulys. “The Solid Fuel Rocket Motor Constructions and Thrust Characteristic Analysis”. Proceedings of 12th International Conference, Mechanica, 2007.

42 Philip G. Hill and Carl R. Peterson. "Mechanics and Thermodynamics of Propulsion". Second Edition, Prentice Hall, November 1992.

43 James S Barrowman. “THE PRACTICAL CALCULATION OF THE AERODYNAMIC CHARACTERISTICS OF SLENDER FINNED VEHICLES”. School of Engineering and Architecture of The Catholic University of America, March 1967.

44 Charles Kittel, Walter D. Knight and Malvin A. Ruderman. “Mechanics”. Berkeley Physics Course Volume1, Second Edition. McGraw-Hill International Book Company, New York. 1973.

45 John D. Anderson, Jr. “Fundamentals of Aerodynamics Sixth Edition”. Mc Graw Hill, 2017.

46 M. Niţă and D. Scholz. “Estimating the Oswald Factor from basic aircraft geometrical parameters”. Deutcher Luft- und Raumfahrtkongress, 2012.

47 Max M. Munk. “General theory of Thin Wing Sections”. National Advisory Committee for Aeronautics, Report no 142. Jan 01, 1923.

48 C. Wayne Dahlke. “THE AERODYNAMIC CHARACTERISTICS OF WRAP-AROUND FINS AT MACH NUMBERS OF 0.3 TO 3.0“. US ARMY MISSILE COMMAND, Redstone Arsenal, Alabama. 15 October 1976.

49 Slobodan Mandić. “Analysis of the Rolling Moment Coefficients of Rockets with Wrap-Around Fins”. Scientific-Technical Review,Vol.LVI,No.2. June 2006.

50 Brett Landon Wilks. “Aerodynamics of Wraparound Fins in Supersonic Flow”. System Simulation and Development Directorate, Aviation and Missile Research, Development and Engineering Centre. April 2006.

51 International Civil Aviation Organization. ” ICAO Standard Atmosphere". ICAO document 7488, 3rd edition, 1993.

52 Mustafa Cavcar. “The International Standard Atmosphere (ISA)”. Anadolu University, 26470, Eskisehir, Turkey.

53 RE Froude. “On the part played in propulsion by differences of fluid pressure”. Trans Inst Naval Architects 1889; 30: 390–405.

54 Jean P Chollet. “CFD actuator disk solutions for a helicopter rotor in hover flight”. University of Manchester Science and Technology, Department of Manufacturing and Aerospace Mechanical Engineering, Sept 2003.

55 Faisal Mahmuddin. “Rotor Blade Performance Analysis with Blade Element Momentum Theory”. Elsevier, The 8th International Conference on Applied Energy – ICAE2016. 2017.

56 Gijsbertus A M van Kuik. “On the limitations of Froude's actuator disc concept”. Technische Universiteit Eindhoven, 01 Jan 1991.

57 L Friedmann, P Ohmer, and M Hajek. “Real-time simulation of rotorcraft downwash in proximity of complex obstacles using grid-based approaches”. 70th Annual Forum of the American Helicopter Society. 2014.

58 Roger Raletz. “Basic theory of the helicopter”. CEPADUES-EDITIONS. ISBN 2-85428.193.4. March 1988.

59 Laurence J Clancy. “Aerodynamics”. Pitman Publishing Limited, London, ISBN 0-273-01120-0. 1975.

60 Shawn Coyle. “Cyclic and Collective”. ISBN 9780557090662, Second Edition, August 26, 2009.

61 Mikael Sjöholm, Nikolas Angelou, Per Hansen, Kasper Hjorth Hansen and Torben Mikkelsen. “Two-Dimensional Rotorcraft Downwash Flow Field Measurements by Lidar-Based Wind Scanners with Agile Beam Steering”. JOURNAL OF ATMOSPHERIC AND OCEANIC TECHNOLOGY, Volume 31, April 2014.

62 Richard K. Moore. “Travelling-Wave Engineering”. McGraw-Hill Book Company, 1960. ISBN 07-042980-4.

63 David F Rogers. “Laminar Flow Analysis”. Cambridge University Press, October 1992. ISBN 0-521-41152-1.

64 Julien C. Thibault and Inanc Senocak. “CUDA Implementation of a Navier-Stokes Solver on Multi-GPU Desktop Platforms for Incompressible Flows”. Boise State University ScholarWorks, January 1, 2009.

65 Dominic D.J. Chandar, Jay Sitaramany and Dimitri Mavriplis. “A Hybrid Multi-GPU/CPU Computational Framework for Rotorcraft Flows on Unstructured Overset Grids”. 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA. June 24-27, 2013.

66 J Bludau, J Rauleder, L Friedmann and M Hajek. “REAL TIME SIMULATION OF ROTOR INFLOW USING A COUPLED FLIGHT DYNAMICS AND FLUID DYNAMICS SIMULATION”. Institute of Helicopter Technology, Technical University of Munich. 2016.

67 Jos Stam. “A Simple Fluid Solver based on the FFT”. Journal of Graphics Tools, Volume 6, Number 2, 43-52. 2001.

68 Jos Stam. “Real-Time Fluid Dynamics for Games”. Proceedings of the Game Developer Conference, March 2003.

69 Jos Stam. “Interacting with Smoke and Fire in Real Time”. Communications of the ACM, Volume 43, Issue 7, 76-83, 2000.

70 J Gordon Leishman. “Principles of HELICOPTER AEROMECHANICS”. Cambridge University Press, 2006. ISBN 0-521-85860-7.

71 Lewis Fry Richardson. “Weather Prediction by Numerical Process”. Cambridge University Press, 1992. ISBN978-0-521-68044-8.

72 Peter Lynch. “The origins of computer weather prediction and climate modelling”. Journal of Computational Physics, Volume 227, Issue 7, Pages 3431-3444, 20 March 2008.

73 N Foster and D Metaxas. “Modeling the Motion of a Hot, Turbulent Gas”. SIGGRAPH 2001 Conference Proceedings, Annual Conference Series, 181-188, August 1997.

74 R Courant, E Isaacson and M Rees. “On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences”. Communication on Pure and Applied Mathematics 5, 243-255, 1952.

75 Alan J. Wadcock, Lindsay A. Ewing, Eduardo Solis, Mark Potsdam and Ganesh Rajagopalan, “Rotorcraft Downwash Flow Field Study to Understand the Aerodynamics of Helicopter Brownout”, American Helicopter Society Southwest Region Technical Specialists Meeting, “Technologies for the Next Generation of Vertical Lift Aircraft”, Dallas-Fort Worth, TX, October 15-17, 2008.

76 Gregory J. Ward, “Measuring and Modeling Anisotropic Reflection”, Lighting Systems Research Group, Lawrence Berkeley Laboratory, July 1992.

77 Kenneth Steiglitz. “An Introduction to Discrete Systems”. John Wiley & Sons, 1967. ISBN 0-471-82097-0.

78 John von Neumann. “First Draft of a Report on the EDVAC”. Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945.

79 Scott Thornton. “What’s the difference between Von-Neumann and Harvard architectures?”. March 2018. https://www.microcontrollertips.com/whats-the-difference-between-von-neumann-and-harvard-architectures/#. Accessed 17 August 2018

80 John W Backus. “Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs”. Turing Award Lecture, 1977.

81 A M Turing. “On Computable Numbers, with an Application to the Entscheidungsproblem”. The Graduate College, Princeton University, New Jersey, USA. November 1936.

82 Karl Rupp. “40 Years of Microprocessor Trend Data”. https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/. Accessed 19 February 2018.

83 Robert H. Dennard, Fritz H. Gaensslen, Hwa-Nien Yu, V. Leo Rideout, Ernest Bassous and Andre R. Leblank“. Design of Ion-Implanted MOSFET’s with Very Small Physical Dimensions”. IEEE JOURNAL OF SOLID-STATE CIRCUITS, vol. SC-9, no. 5, pp. 256–268, October 1974.

84 Gordon E. Moore. “Cramming More Components onto Integrated Circuits”. Electronics, pp. 114–117, April 1965.

85 Ray Kurzweil. “The singularity is near”. Penguin Group, ISBN 0-670-03384-7. 2005.

86 Gene M Amdahl, “Validity of the single processor approach to achieving large scale computing capabilities”, IBM Sunnyvale, California, AFIPS spring joint computer conference, 1967.

87 John L Gustafson. “Reevaluating Amdahl's Law”. Communications of the ACM, Volume 31, May 1988.

88 Michael Galloy. “CPU vs GPU performance”. http://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html#comment-813776. Accessed 19 February 2018.

89 Kayvon Fatahalian and Mike Houston. “A Closer Look at GPUs”. Communications of the ACM, Volume 51, number 10. October 2008.

90 Shane Cook. “CUDA Programming: A Developer's Guide to Parallel Computing with GPUs”. Elsevier. ISBN: 978-0-12-415933-4. 2013.

91 Michael J. Flynn. “Some Computer Organizations and Their Effectiveness”. IEEE Transactions on Computers, Volume: C-21, Issue: 9. September 1972.

92 Gazettabyte. “Altera’s 30 billion transistor FPGA” , 28 June 2015. http://www.gazettabyte.com/home/2015/6/28/alteras-30-billion-transistor-fpga.html. Accessed 25 February 2018.

93 Intel. “Stratix 10”. https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html. Accessed 25 February 2018.

94 Patrick H. Madden. “Parallel Computing: The Elephant in the Room”. SUNY Binghamton Computer Science Department, 19 August 2010.

95 Altera Corporation. “Creating Multiprocessor Nios II Systems”. TU-N2033005-2.0, June 2011.

96 David H. Jones, Adam Powell, Christos-Savvas Bouganis, Peter Y. K. Cheung. “GPU versus FPGA for high productivity computing”. 2010 International Conference on Field Programmable Logic and Applications. 2010.

97 Michael Parker. “Understanding Peak Floating-Point Performance Claims”. Intel White Paper WP-01222-1.1.

98 K. Tsoi and W. Luk. “Axel: A heterogeneous cluster with FPGAs and GPUs”. Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, pp. 115–124, 2010.

99 David R. Martinez, Robert A. Bond, M. Michael Vai. “High Performance Embedded Computing Handbook: A Systems Perspective”. ISBN 978-0-8493-7197-4, CRC Press. Factor 200 improvement - FPGA over software, page 230.

100 Intel. “Designing for Stratix 10 Devices with Power in Mind”. Document AN-767, June 2016.

101 American Department of Defence. “MIL-STD-1553B, MILITARY STANDARD: AIRCRAFT INTERNAL TIME DIVISION COMMAND/RESPONSE MULTIPLEX DATA BUS”. 21 September 1978.

102 American Department of Defence."MIL-STD-1760A, DEPARTMENT OF DEFENSE INTERFACE STANDARD: AIRCRAFT/STORE ELECTRICAL INTERCONNECTION SYSTEM". March 2004.

103 Wikipedia. "ENIAC". http://en.wikipedia.org/wiki/ENIAC. Accessed 23 April 2018.

104 Adrian Landman. "Stores Management and Delivery on CSH-2 Rooivalk: An Integrated Approach". Fourth South African weapon system symposium, Item C17, CSIR, August 1990.

105 Christian Miraglia. "Compression Characteristics of Thermal Interface Gap Filler Materials". Fujipoly. 2018.

106 H. R. Goshayeshi and F. Ampofo. "Heat Transfer by Natural Convection from Vertical and Horizontal Surfaces Using Vertical Fins". Scientific Research, Energy and Power Engineering. November 2009.

107 Intel Corporation. "Intel Stratix 10 Device Datasheet". Intel, S10-DATASHEET, December 2017.

108 A. Khabari, M. Zenouzi, T. O'Connor and A. Rodas. "Natural and Forced Convective Heat Transfer Analysis of Nanostructured Surface". Proceedings of the World Congress on Engineering 2014 Vol I, July 2014.

109 M. Bahrami. "Forced Convection Heat Transfer". Simon Fraser University, ENSC 388 (F09).

110 Randall J. Osczevski. "The Basis of Wind Chill". ARCTIC VOL. 48, NO. 4. December 1995.

111 DR Brown, N Fernandez, JA Dirks and TB Stout. "The Prospects of Alternatives to Vapor Compression Technology for Space Cooling and Food Refrigeration Applications". Pacific Northwest National Laboratory, Richland, Washington. March 2010.

112 "Scientists squeeze more than 1,000 cores on to computer chip", https://phys.org/news/2011-01-scientists-cores-chip.html. Accessed 25 July 2017.

113 Indiana University. "Understanding measures of supercomputer performance and storage system capacity". https://kb.iu.edu/d/apeq#measure-flops. Accessed 30 March 2018.

114 ByteParadigm. "Introduction to I²C and SPI protocols". https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/?/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html. Accessed 01 April 2018.

115 Steven W. Smith. "The Scientist and Engineer's Guide to Digital Signal Processing". California Technical Publishing, ISBN 0-9660176-3-3. 1997.

116 Mark Corless and Arvind Ananthan. "Model-Based Design of Fixed-Point Filters for Embedded Systems". SAE International Journal of Passenger Cars – Electronic and Electrical Systems. October 2009.

117 Leonard E Fuller, "LINEAR ALGEBRA With Applications", Kansas State University, Dickenson Publishing Company, L.C. CAT. CARD NO. 66-13790.

118 Nicholas J Higham, "Accuracy and Stability of Numerical Algorithms", Society for Industrial and Applied Mathematics, Philadelphia, ISBN 0-

89871-355-2.

119 Xin Gui Fang and George Havas, “On the Worst-case Complexity of Integer Gaussian Elimination”, School of Information Technology, University of Queensland, Queensland 4072 Australia, 1997.

120 R. Baur, J. Gläß, F. Klefenz, Ma Long, R. Männer. “THE DEVELOPMENT OF SYSTOLIC PROCESSOR ARRAYS FOR PATTERN RECOGNITION, IMAGE, AND SIGNAL PROCESSING AT THE UNIVERSITIES OF HEIDELBERG AND MANNHEIM - A STATUS REPORT”. ResearchGate. April 1999.

121 Wikipedia. “Interval Arithmetic”. https://en.wikipedia.org/wiki/Interval_arithmetic Accessed 04 September 2017.

122 Ga¨el Deest, Tomofumi Yuki, Olivier Sentieys, Steven Derrien. “Toward Scalable Source Level Accuracy Analysis for Floating-point to Fixed-point Conversion”. 2014 IEEE/ACM International Conference on Computer-Aided Design, San Jose, United States. pp.726–733. November 2014.

123 Darryl D. Lin, Sachin S. Talathi, V. Sreekanth Annapureddy. “Fixed Point Quantization of Deep Convolutional Networks”. arXiv:1511.06393v3 [cs.LG] 2 Jun 2016.

124 Intel. “Find Estimated Critical Paths Without Synthesis Tools”. https://www.mathworks.com/help/hdlcoder/ug/find-estimated-critical-paths-without-synthesis-tools.html?s_tid=srchtitle. Accessed 28 September 2017.

125 Leandro de Souza Rosa and Vanderlei Bonato. “A method to convert floating to fixed-point EKF-SLAM for embedded robotics”. The Brazilian Computer Society. 2012.

126 TheMathworks. “SIMULINK Simulation and Model-Based Design”. http://www.mathworks.com/products/simulink/ Accessed 21 May 2018.

127 Pansaporn Sribonfha, Sirod Sirisup and Pawat Chusilp. “An Efficient Approach for Nozzle Rocket Thrust Estimation”. 2012 IEEE 15th International Conference on Computational Science and Engineering.

128 Adrian Landman. “Stores Management and Delivery on CSH-2 Rooivalk: An Integrated Approach”. Fourth South African weapon system symposium, Item C17, CSIR, August 1990.

129 Debashis Sahoo, Anuraha Annaswamy and Farrukh Alvi. “Active Store Trajectory Control in Supersonic Cavities Using Microjets and Low-Order Modeling”. DOI: 10.2514/1.18007, Massachusetts Institute of Technology, Florida A&M University and Florida State University.

130 Pansaporn Sribonfha, Sirod Sirisup and Pawat Chusilp. “An Efficient Approach for Nozzle Rocket Thrust Estimation”. 2012 IEEE 15th International Conference on Computational Science and Engineering.

131 IBM. “DOORS Next Generation”. https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/doors_next_gen?lang=en Accessed 21 May 2018.

132 Vitech. “Systems Engineering Tools”. http://www.vitechcorp.com/products/core.shtml Accessed 21 May 2018.

133 Rodney K Bogue and Henk W Jentink. “Optical Air Flow Measurements in Flight”. NASA/TP-2004-210735. 2004.

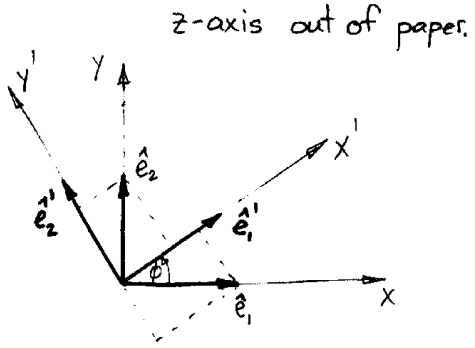
Appendix A
Direction cosine for ZY'X'' transformation

DC4 is a direction cosine matrix for transforming a vector from co-ordinate system 2 into co-ordinate system 1 where both co-ordinate systems are right-handed and co-ordinate system 2 is defined by rotation of co-ordinate system 1 through the three Euler angles ϕ, θ and ψ around the Z, Y' and X'' axes in the same order. Starting with unit vectors $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$ and $\hat{\mathbf{e}}_3$ along the X, Y and Z-axes:

Rotation through angle ϕ around Z-axis.

$$\begin{aligned} \hat{e}_1 &= \cos(\phi)\hat{e}_1' - \sin(\phi)\hat{e}_2' + 0\hat{e}_3' \\ \hat{e}_2 &= \sin(\phi)\hat{e}_1' + \cos(\phi)\hat{e}_2' + 0\hat{e}_3' \\ \hat{e}_3 &= 0\hat{e}_1' + 0\hat{e}_2' + 1\hat{e}_3' \end{aligned} \qquad (1)$$

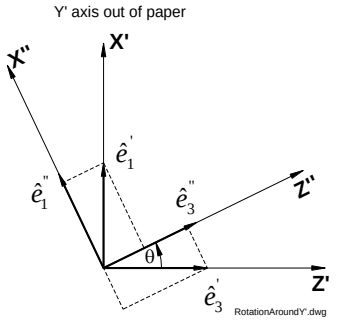
$$or: \begin{bmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{e}_1' \\ \hat{e}_2' \\ \hat{e}_3' \end{bmatrix}$$



Rotation through angle θ around Y'-axis.

$$\begin{aligned} \hat{e}_1' &= \cos(\theta)\hat{e}_1'' + 0\hat{e}_2'' + \sin(\theta)\hat{e}_3'' \\ \hat{e}_2' &= 0\hat{e}_1'' + 1\hat{e}_2'' + 0\hat{e}_3'' \\ \hat{e}_3' &= -\sin(\theta)\hat{e}_1'' + 0\hat{e}_2'' + \cos(\theta)\hat{e}_3'' \end{aligned} \qquad (2)$$

$$or: \begin{bmatrix} \hat{e}_1' \\ \hat{e}_2' \\ \hat{e}_3' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \hat{e}_1'' \\ \hat{e}_2'' \\ \hat{e}_3'' \end{bmatrix}$$



Rotation through angle ψ around X''-axis.

$$\begin{aligned} \hat{e}_1'' &= 1\hat{e}_1''' + 0\hat{e}_2''' + 0\hat{e}_3''' \\ \hat{e}_2'' &= 0\hat{e}_1''' + \cos(\psi)\hat{e}_2''' - \sin(\psi)\hat{e}_3''' \\ \hat{e}_3'' &= 0\hat{e}_1''' + \sin(\psi)\hat{e}_2''' + \cos(\psi)\hat{e}_3''' \end{aligned} \qquad (3)$$

$$or: \begin{bmatrix} \hat{e}_1'' \\ \hat{e}_2'' \\ \hat{e}_3'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} \hat{e}_1''' \\ \hat{e}_2''' \\ \hat{e}_3''' \end{bmatrix}$$

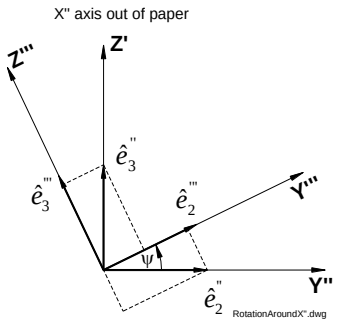


Figure 1. Rotations of the DC4 direction cosine matrix.

Equation (2) into equation (1) gives:

$$\begin{bmatrix}\hat{e}_1\\ \hat{e}_2\\ \hat{e}_3\end{bmatrix} = \begin{bmatrix}\cos(\phi) & -\sin(\phi) & 0\\ \sin(\phi) & \cos(\phi) & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}\cdot\begin{bmatrix}\hat{e}_1^{''}\\ \hat{e}_2^{''}\\ \hat{e}_3^{''}\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix}\hat{e}_1\\ \hat{e}_2\\ \hat{e}_3\end{bmatrix} = \begin{bmatrix}\cos(\phi)\cos(\theta) & -\sin(\phi) & \cos(\phi)\sin(\theta)\\ \sin(\phi)\cos(\theta) & \cos(\phi) & \sin(\phi)\sin(\theta)\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}\cdot\begin{bmatrix}\hat{e}_1^{''}\\ \hat{e}_2^{''}\\ \hat{e}_3^{''}\end{bmatrix} \quad (4)$$

Equation (3) into equation (4) gives:

$$\begin{bmatrix}\hat{e}_1\\ \hat{e}_2\\ \hat{e}_3\end{bmatrix} = \begin{bmatrix}\cos(\phi)\cos(\theta) & -\sin(\phi) & \cos(\phi)\sin(\theta)\\ \sin(\phi)\cos(\theta) & \cos(\phi) & \sin(\phi)\sin(\theta)\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}\cdot\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\psi) & -\sin(\psi)\\ 0 & \sin(\psi) & \cos(\psi)\end{bmatrix}\cdot\begin{bmatrix}\hat{e}_1^{'''}\\ \hat{e}_2^{'''}\\ \hat{e}_3^{'''}\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix}\hat{e}_1\\ \hat{e}_2\\ \hat{e}_3\end{bmatrix} = \begin{bmatrix}\cos(\phi)\cos(\theta) & -\sin(\phi)\cos(\psi)+\cos(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\sin(\psi)+\cos(\phi)\sin(\theta)\cos(\psi)\\ \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\theta)\sin(\psi) & -\cos(\theta)\sin(\psi)+\sin(\phi)\sin(\theta)\cos(\psi)\\ -\sin(\theta) & \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi)\end{bmatrix}\cdot\begin{bmatrix}\hat{e}_1^{'''}\\ \hat{e}_2^{'''}\\ \hat{e}_3^{'''}\end{bmatrix} \quad (5)$$

$$\Rightarrow \begin{bmatrix}\hat{e}_1\\ \hat{e}_2\\ \hat{e}_3\end{bmatrix} = \mathbf{DC}4\cdot\begin{bmatrix}\hat{e}_1^{'''}\\ \hat{e}_2^{'''}\\ \hat{e}_3^{'''}\end{bmatrix}$$

Appendix B
The RigidElement2 block

The **RigidElement2** block models a rigid body with arbitrary shape that is subject to translation and rotation as a result of two sets of forces and torques acting through two "connecting nodes" located at arbitrary positions on or within the rigid body. In addition, it assumes a force and torque to act through the Center of Mass of the rigid body (such as gravity that depends on the position of the Center of Mass of the rigid body in inertial space).

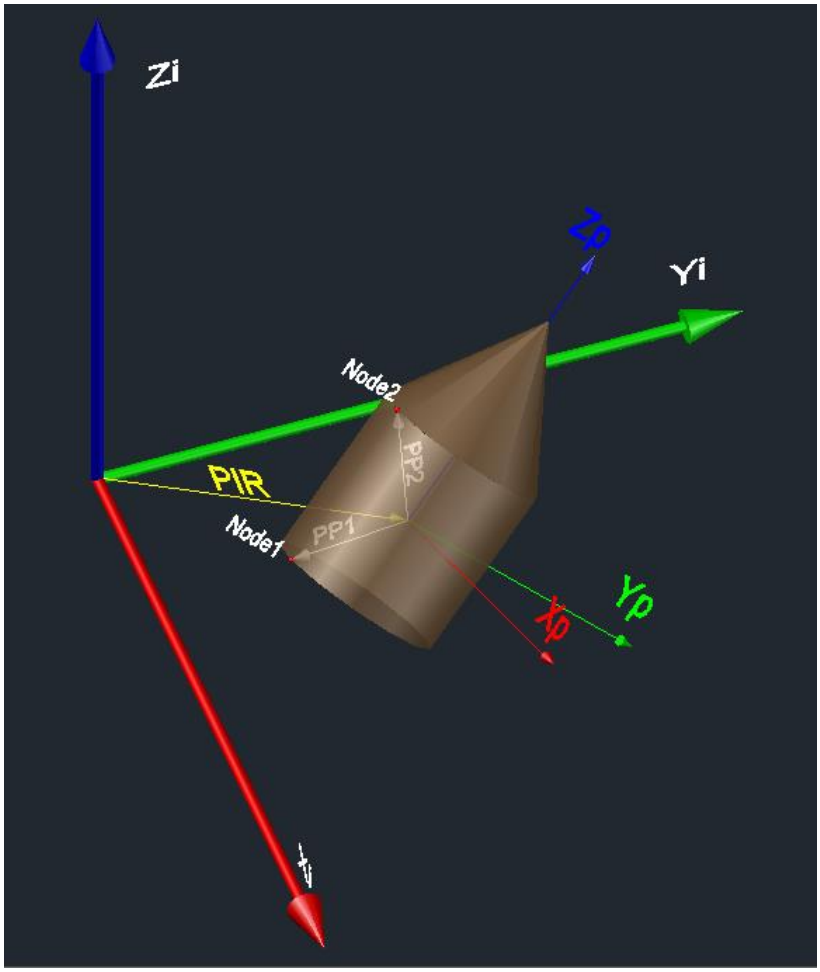


Figure 1. Inertial and principal axes for describing the motion of a rigid body.

To develop the ***RigidElement2*** model, consider Figure 1 which illustrates a rigid body at some or other arbitrary position and orientation in space. Assume the rigid body to have a set of principal axes with origin at the Center of Mass of the rigid body. Assume that initially the principal axes were aligned with inertial axes, but that the rigid body has translated and rotated through the three Euler angles ϕ_r, θ_r and ψ_r in a Z-X'-Y'' rotation scheme. Also assume the position of the Center of Mass of the rigid body to be defined by vector **PIR** that describes the translation of the rigid body in terms of inertial axes. In contrast, assume the positions of the two Connecting Nodes to be described by vectors **PP1** and **PP2** which are defined in terms of the principal axes of the Rigid Body.

From Figure 1 we note that the vector sum of the force acting through the Center of Mass and the two forces acting through the Connecting Nodes will cause the Center of Mass of the rigid body to move in accordance with the law for the conservation of linear momentum. At the same

time, the components of the two forces acting orthogonal to vectors **PP1** and **PP2** will generate torques that will combine with the torque acting on the Center of Mass and the two torques acting on the Connecting Nodes to cause the Rigid Body to rotate around its Center of Mass in accordance with the law for the conservation of angular momentum. To describe these effects mathematically, we start by writing for the angular velocity $\overline{\omega}$ of the Rigid Body in terms of its principal axes:

$$\overline{\omega} = \omega_1 \hat{\mathbf{e}}_1 + \omega_2 \hat{\mathbf{e}}_2 + \omega_3 \hat{\mathbf{e}}_3 \qquad \textbf{Equation 127}$$

where $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$ are unit vectors along the Xp, Yp and Zp principal axes while ω_1, ω_2 and ω_3 are the components of the angular velocity of the Rigid Body in principal axes. For a set of principal axes the products of inertia are zero so that we can write for the angular momentum ***JPR*** of the Rigid Body in principal axes:

$$\mathbf{JPR} = I_1 \omega_1 \hat{\mathbf{e}}_1 + I_2 \omega_2 \hat{\mathbf{e}}_2 + I_3 \omega_3 \hat{\mathbf{e}}_3 \qquad \textbf{Equation 128}$$

where I_1, I_2 and I_3 represent the Moments of Inertia of the Rigid Body around the Xp, Yp and Zp principal axes. From the law of conservation of angular momentum, the rate of change in angular momentum of the rigid body must be equal to the external torque ***NPR*** acting on it:

$$\mathbf{NPR} = \left.\frac{d\mathbf{JPR}}{dt}\right|_{inertial} = \left.\frac{d\mathbf{JPR}}{dt}\right|_{principal} + \overline{\omega} \times \mathbf{JPR} \qquad \textbf{Equation 129}$$

Differentiation of Equation 128, evaluation of the cross product between Equation 127 and Equation 128, and substitution of the results in Equation 129 gives:

$$\mathbf{NPR} = \left[I_1 \dot{\omega}_1 + (I_3 - I_2)\omega_2 \omega_3\right]\hat{\mathbf{e}}_1 + \left[I_2 \dot{\omega}_2 + (I_1 - I_3)\omega_1 \omega_3\right]\hat{\mathbf{e}}_2 + \left[I_3 \dot{\omega}_3 + (I_2 - I_1)\omega_2 \omega_3\right]\hat{\mathbf{e}}_3$$

$$\Rightarrow \begin{cases} NPR_1 &= I_1 \dot{\omega}_1 + (I_3 - I_2)\omega_2 \omega_3 \\ NPR_2 &= I_2 \dot{\omega}_2 + (I_1 - I_3)\omega_3 \omega_1 \\ NPR_3 &= I_3 \dot{\omega}_3 + (I_2 - I_1)\omega_1 \omega_2 \end{cases} \qquad \textbf{Equation 130}$$

where NPR_1, NPR_2 and NPR_3 are the components of external torque exerted on the rigid body along the Xp, Yp and Zp principal axes. Note that Equation 130 is known as Euler's equation of rotary motion. From the direction cosine equations for rotating a vector from one coordinate system into another, we can derive expressions for ω_1, ω_2 and ω_3 in terms of ϕ_r, θ_r and ψ_r. Hence, we note that a change in angle ϕ_r results in an angular velocity $\overline{\omega}_{phi}$ along the Zp' axis. Using the first equation of Z-X'-Y'' rotation scheme with $\theta_r = 0$ and $\psi_r = 0$ to transform this vector into inertial co-ordinates we find:

$$\begin{aligned} \overline{\omega}_{phi} &= \begin{bmatrix} \cos(\phi_r) & -\sin(\phi_r) & 0 \\ \sin(\phi_r) & \cos(\phi_r) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\phi}_r \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ \dot{\phi}_r \end{bmatrix} \end{aligned} \qquad \textbf{Equation 131}$$

Similarly, a change in angle θ_r results in an angular velocity $\overline{\omega}_{theta}$ along the Xp'' axis. Using the second equation of Z-X'-Y'' rotation scheme with $\psi_r = 0$ to transform this vector into inertial co-ordinates we find:

$$\begin{aligned}\overline{\omega}_{theta} &= \begin{bmatrix}\cos(\phi_r) & -\sin(\phi_r)\cos(\theta_r) & \sin(\phi_r)\sin(\theta_r)\\ \sin(\phi_r) & \cos(\phi_r)\cos(\theta_r) & -\cos(\phi_r)\sin(\theta_r)\\ 0 & \sin(\theta_r) & \cos(\theta_r)\end{bmatrix}\cdot\begin{bmatrix}\dot{\theta}_r\\ 0\\ 0\end{bmatrix}\\ &= \begin{bmatrix}\dot{\theta}_r\cos(\phi_r)\\ \dot{\theta}_r\sin(\phi_r)\\ 0\end{bmatrix}\end{aligned}$$

Equation 132

Similarly, a change in angle ψ_r results in an angular velocity $\overline{\omega}_{psi}$ along the Yp''' axis. Using the third equation of Z-X'-Y'' rotation scheme to transform this vector into inertial co-ordinates we find:

$$\begin{aligned}\overline{\omega}_{psi} &= \begin{bmatrix}\cos(\phi_r)\cos(\psi_r)-\sin(\phi_r)\sin(\theta_r)\sin(\psi_r) & -\sin(\phi_r)\cos(\theta_r) & \cos(\phi_r)\sin(\psi_r)+\sin(\phi_r)\sin(\theta_r)\cos(\psi_r)\\ \sin(\phi_r)\cos(\psi_r)+\cos(\phi_r)\sin(\theta_r)\sin(\psi_r) & \cos(\phi_r)\cos(\theta_r) & \sin(\phi_r)\sin(\psi_r)-\cos(\phi_r)\sin(\theta_r)\cos(\psi_r)\\ -\cos(\theta_r)\sin(\psi_r) & \sin(\theta_r) & \cos(\theta_r)\cos(\psi_r)\end{bmatrix}\cdot\begin{bmatrix}0\\ \dot{\psi}_r\\ 0\end{bmatrix}\\ &= \begin{bmatrix}-\dot{\psi}_r\sin(\phi_r)\cos(\theta_r)\\ \dot{\psi}_r\cos(\phi_r)\cos(\theta_r)\\ \dot{\psi}_r\sin(\theta_r)\end{bmatrix}\end{aligned}$$

Equation 133

so that the total angular velocity $\overline{\omega}$ of the Rigid Body in inertial axes is given by:

$$\begin{aligned}\overline{\omega} &= \overline{\omega}_{phi}+\overline{\omega}_{theta}+\overline{\omega}_{psi}\\ &\equiv \begin{bmatrix}\dot{\theta}_r\cos(\phi_r)-\dot{\psi}_r\sin(\phi_r)\cos(\theta_r)\\ \dot{\theta}_r\sin(\phi_r)+\dot{\psi}_r\cos(\phi_r)\cos(\theta_r)\\ \dot{\phi}_r+\dot{\psi}_r\sin(\theta_r)\end{bmatrix}\end{aligned}$$

Equation 134

However, Equation 129 requires $\overline{\omega}$ to be expressed in principal axes. Hence, transforming $\overline{\omega}$ from principal axes into inertial axes with the use of Z-X'-Y'' rotation scheme and equating the result to Equation 134 we have:

$$\begin{bmatrix}\dot{\theta}_r\cos(\phi_r)-\dot{\psi}_r\sin(\phi_r)\cos(\theta_r)\\ \dot{\theta}_r\sin(\phi_r)+\dot{\psi}_r\cos(\phi_r)\cos(\theta_r)\\ \dot{\phi}_r+\dot{\psi}_r\sin(\theta_r)\end{bmatrix} = \boldsymbol{DC}\cdot\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \boldsymbol{DC}^{-1}\cdot\begin{bmatrix}\dot{\theta}_r\cos(\phi_r)-\dot{\psi}_r\sin(\phi_r)\cos(\theta_r)\\ \dot{\theta}_r\sin(\phi_r)+\dot{\psi}_r\cos(\phi_r)\cos(\theta_r)\\ \dot{\phi}_r+\dot{\psi}_r\sin(\theta_r)\end{bmatrix}$$

Equation 135

$$\Rightarrow \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}\dot{\theta}_r\cos(\psi_r)-\dot{\phi}\cos(\theta_r)\sin(\psi_r)\\ \dot{\psi}_r+\dot{\phi}_r\sin(\theta_r)\\ \dot{\theta}_r\sin(\psi_r)+\dot{\phi}_r\cos(\theta_r)\cos(\psi_r)\end{bmatrix}$$

where we have made use of the inverse direction cosine matrix. Differentiation of Equation 135 gives:

$$
\begin{aligned}
\dot{\omega}_1 &= \ddot{\theta}_r \cos(\psi_r) - \ddot{\phi}_r \cos(\theta_r)\sin(\psi_r) - \dot{\theta}_r\dot{\psi}_r \sin(\psi_r) + \\
&\quad \dot{\phi}_r\dot{\theta}_r \sin(\theta_r)\sin(\psi_r) - \dot{\phi}_r\dot{\psi}_r \cos(\theta_r)\cos(\psi_r) \\
\dot{\omega}_2 &= \ddot{\psi}_r + \ddot{\phi}_r \sin(\theta_r) + \dot{\phi}_r\dot{\theta}_r \cos(\theta_r) \\
\dot{\omega}_3 &= \ddot{\theta}_r \sin(\psi_r) + \dot{\theta}_r\dot{\psi}_r \cos(\psi_r) + \ddot{\phi}_r \cos(\theta_r)\cos(\psi_r) - \\
&\quad \dot{\phi}_r\dot{\theta}_r \sin(\theta_r)\cos(\psi_r) - \dot{\phi}_r\dot{\psi}_r \cos(\theta_r)\sin(\psi_r)
\end{aligned}
$$

Equation 136

Substituting ω_2 and ω_3 as given by Equation 135 and $\dot{\omega}_1$ as given by Equation 136 into the first term of Equation 130 and introducing simplification terms A_{11}, A_{12}, A_{13} and A_{14} gives:

$$
\begin{aligned}
&\ddot{\phi}_r A_{11} + \ddot{\theta}_r A_{12} + \ddot{\psi}_r A_{13} = NPR_1 - A_{14} \\
&\textit{with} \quad A_{11} = -I_1 \cos(\theta_r)\sin(\psi_r) \\
&\qquad A_{12} = I_1 \cos(\psi_r) \\
&\qquad A_{13} = 0 \\
&\qquad A_{14} = I_1\left\{-\dot{\theta}_r\dot{\psi}_r \sin(\psi_r) + \dot{\phi}_r\dot{\theta}_r \sin(\theta_r)\sin(\psi_r) - \dot{\phi}_r\dot{\psi}_r \cos(\theta_r)\cos(\psi_r)\right\} + \\
&\qquad\qquad (I_3 - I_2)\left\{\dot{\psi}_r + \dot{\phi}_r \sin(\theta_r)\right\}\left\{\dot{\theta}_r \sin(\psi_r) + \dot{\phi}_r \cos(\theta_r)\cos(\psi_r)\right\}
\end{aligned}
$$

Equation 137

Substituting ω_3 and ω_1 as given by Equation 135 and $\dot{\omega}_2$ as given by Equation 136 into the second term of Equation 130 and introducing simplification terms A_{21}, A_{22}, A_{23} and A_{24} gives:

$$
\begin{aligned}
&\ddot{\phi}_r A_{21} + \ddot{\theta}_r A_{22} + \ddot{\psi}_r A_{23} = NPR_2 - A_{24} \\
&\textit{with} \quad A_{21} = I_2 \sin(\theta_r) \\
&\qquad A_{22} = 0 \\
&\qquad A_{23} = I_2 \\
&\qquad A_{24} = I_2\dot{\phi}_r\dot{\theta}_r \cos(\theta_r) + \\
&\qquad\qquad (I_1 - I_3)\left\{\dot{\theta}_r \sin(\psi_r) + \dot{\phi}_r \cos(\theta_r)\cos(\psi_r)\right\}\left\{\dot{\theta}_r \cos(\psi_r) - \dot{\phi}_r \cos(\theta_r)\sin(\psi_r)\right\}
\end{aligned}
$$

Equation 138

Substituting ω_1 and ω_2 as given by Equation 135 and $\dot{\omega}_3$ as given by Equation 136 into the third term of Equation 130 and introducing simplification terms A_{31}, A_{32}, A_{33} and A_{34} gives:

$$
\begin{aligned}
&\ddot{\phi}_r A_{31} + \ddot{\theta}_r A_{32} + \ddot{\psi}_r A_{33} = NPR_3 - A_{34} \\
&\textit{where}: \quad A_{31} = I_3 \cos(\theta_r)\cos(\psi_r) \\
&\qquad A_{32} = I_3 \sin(\psi_r) \\
&\qquad A_{33} = 0 \\
&\qquad A_{34} = I_3\left\{\dot{\theta}_r\dot{\psi}_r \cos(\psi_r) - \dot{\phi}_r\dot{\theta}_r \sin(\theta_r)\cos(\psi_r) - \dot{\phi}_r\dot{\psi}_r \cos(\theta_r)\sin(\psi_r)\right\} + \\
&\qquad\qquad (I_2 - I_1)\left\{\dot{\theta}_r \cos(\psi_r) - \dot{\phi}_r \cos(\theta_r)\sin(\psi_r)\right\}\left\{\dot{\psi}_r + \dot{\phi}_r \sin(\theta_r)\right\}
\end{aligned}
$$

Equation 139

Hence, we have three equations in three unknowns. Solving for $\ddot{\phi}_r$ with Kramer's rule and using the fact that A_{13}, A_{22} and A_{33} are all zero, we find:

$$\ddot{\phi}_r = \frac{\begin{vmatrix} (NPR_1 - A_{14}) & A_{12} & A_{13} \\ (NPR_2 - A_{24}) & A_{22} & A_{23} \\ (NPR_3 - A_{34}) & A_{32} & A_{33} \end{vmatrix}}{\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}} \qquad \textbf{Equation 140}$$

$$= \frac{A_{12}(NPR_3 - A_{34}) - A_{32}(NPR_1 - A_{14})}{A_{12}A_{31} - A_{11}A_{32}}$$

Solving for $\ddot{\theta}_r$ in similar fashion:

$$\ddot{\theta}_r = \frac{\begin{vmatrix} A_{11} & (NPR_1 - A_{14}) & A_{13} \\ A_{21} & (NPR_2 - A_{24}) & A_{23} \\ A_{31} & (NPR_3 - A_{34}) & A_{33} \end{vmatrix}}{\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}} \qquad \textbf{Equation 141}$$

$$= \frac{A_{31}(NPR_1 - A_{14}) - A_{11}(NPR_3 - A_{34})}{A_{12}A_{31} - A_{11}A_{32}}$$

Solving for $\ddot{\psi}_r$ in similar fashion:

$$\ddot{\psi}_r = \frac{\begin{vmatrix} A_{11} & A_{12} & (NPR_1 - A_{14}) \\ A_{21} & A_{22} & (NPR_2 - A_{24}) \\ A_{31} & A_{32} & (NPR_3 - A_{34}) \end{vmatrix}}{\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}} \qquad \textbf{Equation 142}$$

$$= \frac{A_{21}A_{32}(NPR_1 - A_{14}) + (A_{12}A_{31} - A_{11}A_{32})(NPR_2 - A_{24}) - A_{12}A_{21}(NPR_3 - A_{34})}{A_{23}(A_{12}A_{31} - A_{11}A_{32})}$$

We now have three differential equations in ϕ_r, θ_r and ψ_r of a form suitable for solving with the ODE solvers of Matlab.

8.19.8 Avoiding division with zero errors

The denominator of Equation 140, Equation 141 and Equation 142 needs further investigation. To ensure correct operation of the algorithm, this denominator must never be zero – a condition for which Kramer's rule would become undefined. Substituting the original expressions into the denominator and noting that $A_{23} = I_y$ (i.e. a constant), we find:

$$\begin{aligned} A_{12}A_{31} - A_{11}A_{32} &= I_1 \cos(\psi_r) I_3 \cos(\theta_r)\cos(\psi_r) + I_1 \cos(\theta_r)\sin(\psi_r) I_3 \sin(\psi_r) \\ &= I_1 I_3 \cos(\theta_r)\cos^2(\psi_r) + I_1 I_3 \cos(\theta_r)\sin^2(\psi_r) \\ &= -I_1 I_3 \cos(\theta_r) \end{aligned} \qquad \textbf{Equation 143}$$

It is clear from Equation 143 that the denominator of Equation 140, Equation 141 and Equation 142 will become zero when $cos(\theta_r) = 0$. In other words for $\theta_r = (2n+1)\pi/2$ with n any integer - a condition which is bound to occur. Hence this condition must be checked for and a different set of equations used to determine $\ddot{\phi}_r$, $\ddot{\theta}_r$ and $\ddot{\psi}_r$ when it occurs. Fortunately Equation 137, Equation 138 and Equation 139 reduce to relatively simple equations under these circumstances. There are two cases:

8.19.8.1 ψ_r not equal to an uneven multiple of $\pi/2$

To find an expression for $\ddot{\theta}_r$ under these conditions we substitute θ_r with $\pi/2$ in Equation 137 and we find:

$$\begin{aligned}
&\ddot{\theta}_r I_1 \cos(\psi_r) = NPR_1 - I_1\left[-\dot{\theta}_r\dot{\psi}_r \sin(\psi_r) + \dot{\phi}_r\dot{\theta}_r \sin(\psi_r)\right] + (I_3 - I_2)\left[\dot{\psi}_r + \dot{\phi}_r\right]\left[\dot{\theta}_r \sin(\psi_r)\right] \\
\Rightarrow\quad & NPR_1 = I_1\left[\ddot{\theta}_r \cos(\psi_r) - \dot{\theta}_r\dot{\psi}_r \sin(\psi_r) + \dot{\phi}_r\dot{\theta}_r \sin(\psi_r)\right] + (I_3 - I_2)\left[\dot{\psi}_r + \dot{\phi}_r\right]\left[\dot{\theta}_r \sin(\psi_r)\right] \\
\Rightarrow\quad & \ddot{\theta}_r \cos(\psi_r) - \dot{\theta}\sin(\psi_r)\left[\dot{\psi}_r - \dot{\phi}\right] = \frac{NPR_1 - (I_3 - I_2)\left[\dot{\psi}_r + \dot{\phi}_r\right]\left[\dot{\theta}_r \sin(\psi_r)\right]}{I_1} \\
\Rightarrow\quad & \ddot{\theta}_r \cos(\psi_r) = \frac{NPR_1 - (I_3 - I_2)\left[\dot{\psi}_r + \dot{\phi}_r\right]\left[\dot{\theta}_r \sin(\psi_r)\right]}{I_1} + \frac{\dot{\theta}\sin(\psi_r)\left[\dot{\psi}_r - \dot{\phi}\right]I_1}{I_1}
\end{aligned}$$ **Equation 144**

$$\therefore\quad \ddot{\theta}_r = \left\{ \begin{array}{l} \dfrac{NPR_1 - \dot{\theta}\sin(\psi_r)\left[(I_3 - I_2)(\dot{\psi}_r + \dot{\phi}_r) - I_1(\dot{\psi}_r - \dot{\phi})\right]}{I_1 \cos(\psi_r)} \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r \neq (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$

To find an expression for $\ddot{\phi}_r$ and $\ddot{\psi}_r$ we substitute θ_r with $\pi/2$ in Equation 138 and we find:

$$\begin{aligned}
&\ddot{\phi}_r I_2 + \ddot{\psi}_r I_2 = NPR_2 - (I_1 - I_3)\left[\dot{\theta}_r \sin(\psi_r)\right]\left[\dot{\theta}_r \cos(\psi_r)\right] \\
\Rightarrow\quad & NPR_2 = I_2\left[\ddot{\phi}_r + \ddot{\psi}_r\right] + (I_1 - I_3)\left[\dot{\theta}_r \sin(\psi_r)\right]\left[\dot{\theta}_r \cos(\psi_r)\right]
\end{aligned}$$

$$\therefore\quad \ddot{\phi}_r + \ddot{\psi}_r = \left\{ \begin{array}{l} \dfrac{NPR_2 - (I_1 - I_3)\dot{\theta}_r^2 \sin(\psi_r)\cos(\psi_r)}{I_2} \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r \neq (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 145**

This equation makes perfect sense because when $\cos(\theta_r) = 0$ the Yp''' and Zi axes align, in which case a rotation along the Z axis can be taken up by changes in either ϕ_r or ψ_r. Since the mathematics does not dictate a choice, assume:

$$\ddot{\phi}_r = 0 \quad with\ \theta_r = (2n+1)\frac{\pi}{2};\ \psi_r \neq (2n+1)\frac{\pi}{2};\ n = \text{int}$$ **Equation 146**

which leads to:

$$\ddot{\psi}_r = \left\{ \begin{array}{l} \dfrac{NPR_2 - (I_1 - I_3)\dot{\theta}_r^2 \sin(\psi_r)\cos(\psi_r)}{I_2} \\ \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r \neq (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 147**

8.19.8.2 ψ_r equal to an uneven multiple of $\pi/2$

For this case Equation 137, Equation 138 and Equation 139 simplifies even further. Since Equation 144 was derived from Equation 137 and contains a $cos(\psi_r)$ term in the denumerator, Equation 137 cannot be used in this case. However, substituting $\pi/2$ for both θ_r and ψ_r in Equation 139 we find:

$$\ddot{\theta}_r I_3 = NPR_3$$

$$\therefore\ \ddot{\theta}_r = \left\{ \begin{array}{l} \dfrac{NPR_3}{I_3} \\ \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r = (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 148**

which makes sense, since with both θ_r and ψ_r equal to $\pi/2$ the Zp''' and Xi axes align. Under such conditions a torque along the Zp''' axis would thus result in angular acceleration around the Xi axis (i.e. acceleration in θ_r). Substituting $\pi/2$ for both θ_r and ψ_r in Equation 138 we find:

$$\ddot{\phi}_r I_2 + \ddot{\psi}_r I_2 = NPR_2$$

$$\therefore\ \ddot{\phi}_r + \ddot{\psi}_r = \left\{ \begin{array}{l} \dfrac{NPR_2}{I_2} \\ \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r = (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 149**

This equation makes sense, since even with both θ_r and ψ_r equal to $\pi/2$ the Yp''' and Zi axes still align, so that a rotation along the Zi axis can again be taken up by changes in either ϕ_r or ψ_r. The mathematics again does not dictate a choice, so assume:

$$\ddot{\phi}_r = \left\{ \begin{array}{l} 0 \\ \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r = (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 150**

which leads to:

$$\ddot{\psi}_r = \left\{ \begin{array}{l} \dfrac{NPR_2}{I_2} \\ \\ with\ \theta_r = (2n+1)\dfrac{\pi}{2};\ \psi_r = (2n+1)\dfrac{\pi}{2};\ n = Integer \end{array} \right\}$$ **Equation 151**

This equation makes sense, since with both θ_r and ψ_r equal to $\pi/2$ the *Yp*''' and *Zi* axes align. Hence, a torque along the *Yp*''' axis would result in angular acceleration around the *Zi* axis (i.e. acceleration in ψ_r if we clamp the acceleration in θ_r at zero).

8.19.9 Calculating External Torque acting on the Rigid Body

We now have a set of equations which describe the rotary motion of the Rigid Body around its Center of Mass as a result of an external torque ***NPR*** acting on the Rigid Body. To calculate this torque, we note that each of the two forces acting through the two Connecting Nodes can be decomposed into two components – one through the Center of Mass of the Rigid Body and one tangential to a line formed between the Center of Mass of the Rigid Body and the point on the Rigid Body through which the force act. It immediately follows that the components working through the Center of Mass of the Rigid Body cannot produce any torques on the Rigid Body, since their moment arms are zero. In contrast, the tangential components do produce torques, the size and direction of which can be calculated as the cross-product between the force and the position vector of the Connecting Node through which it acts. Note that both vectors must be expressed in principal axes for this to hold true. Hence, we can write:

$$\mathbf{NPR} = \mathbf{PP1}\times\mathbf{FP1}+\mathbf{PP2}\times\mathbf{FP2}+\mathbf{NP1}+\mathbf{NP2}+\mathbf{NPC}$$

$$\Rightarrow\begin{cases} NPR_x &= \left[\left(PP1_y FP1_z - PP1_z FP1_y\right)+\left(PP2_y FP2_z - PP2_z FP2_y\right)+NP1_x+NP2_x+NPC_x\right] \\ NPR_y &= \left[\left(PP1_z FP1_x - PP1_x FP1_z\right)+\left(PP2_z FP2_x - PP2_x FP2_z\right)+NP1_y+NP2_y+NPC_y\right] \\ NPR_z &= \left[\left(PP1_x FP1_y - PP1_y FP1_x\right)+\left(PP2_x FP2_y - PP2_y FP2_x\right)+NP1_z+NP2_z+NPC_z\right] \end{cases} \quad \textbf{Equation 152}$$

with ***NPR*** the total torque acting on the Rigid Body, ***NPC*** the torque acting on the Center of Mass, ***NP1*** and ***NP2*** the two torques acting on the two Connecting Nodes and ***FP1*** and ***FP2*** the two forces acting on the two Connecting Nodes, all in principal axes. ***PP1*** and ***PP2*** represent the position vectors of the two Connecting Nodes, also in principal axes. Note that since the Connecting Nodes are stationary points on the Rigid Body, ***PP1*** and ***PP2*** are both constant.

8.19.10 Calculating the Linear Motion of the Rigid Body

To derive an equation for the linear motion of the Center of Mass of the Rigid Body we turn to Newton's second law of motion:

$$\begin{aligned} \frac{d^2}{dt^2}PIR_x &= \left\{\frac{FIR_x}{m_r}\right\} \\ \frac{d^2}{dt^2}PIR_y &= \left\{\frac{FIR_y}{m_r}\right\} \\ \frac{d^2}{dt^2}PIR_z &= \left\{\frac{FIR_z}{m_r}\right\} \end{aligned} \quad \textbf{Equation 153}$$

where PIR_x, PIR_y and PIR_z represent the components of the position vector of the Center of Mass of the Rigid Body in inertial axes. FIR_x, FIR_y and FIR_z represent the components of the total force acting on the Rigid Body in Inertial Axes. Note that the total force acting on the Rigid Body is simply the vector sum of the force acting on the Center of Mass and the two forces acting on the two Connecting Nodes, so that we have:

$$\begin{aligned} FIR_x &= FIC_x + FI2_x + FI2_x \\ FIR_y &= FIC_y + FI2_y + FI2_y \\ FIR_z &= FIC_z + FI2_z + FI2_z \end{aligned} \quad \textbf{Equation 154}$$

with ***FIR*** the total force acting on the Rigid Body, ***FIC*** the force acting on the Center of Mass, ***FI1*** the force acting on Connecting Node 1 and ***FI2*** the force acting on Connecting Node 2, all in Inertial Axes.

8.19.11 Outputs of the Rigid Body model

We now have a set of equations describing the rotational and translational behavior of the Rigid Body as a result of the torques and forces acting on the Center of Mass and the two Connecting Nodes. The outputs from the model, however, must be the mass of the Rigid Body and the rotational and translational states of the Center of Mass and the two Connecting Nodes of the Rigid Body. Since the Rigid Body cannot bend or deform, it follows immediately that the rotational states of the Center of Mass and the two Connecting Nodes must be the same, so that:

$$\begin{aligned} \phi_1 &= \phi_r \\ \phi_2 &= \phi_r \\ \theta_1 &= \theta_r \\ \theta_2 &= \theta_r \\ \psi_1 &= \psi_r \\ \psi_2 &= \psi_r \end{aligned}$$

Equation 155

where ϕ_r, θ_r and ψ_r represent the Euler angles of the Center of Mass while ϕ_1, θ_1 and ψ_1 represent the Euler angles of Connecting Node 1 and ϕ_2, θ_2 and ψ_2 represent the Euler angles of Connecting Node 2.

The ***RigidBody2Ver2.m*** forms part of a SIMULINK sub-system named ***RigidElement2*** with a structure as illustrated in Figure 2. Note that the ***RigidElement2*** block is intended to model a rigid body with two Connecting Nodes. Some cases will be analyzed where rigid bodies with more than two Connecting Nodes are required. The structure and software of these models are the same as the **RigidElement2** block except for a small amount of additional software to cater for the increase of Connecting Nodes as required. This additional code is so simple that these blocks are simply verified through inspection of the additional software.

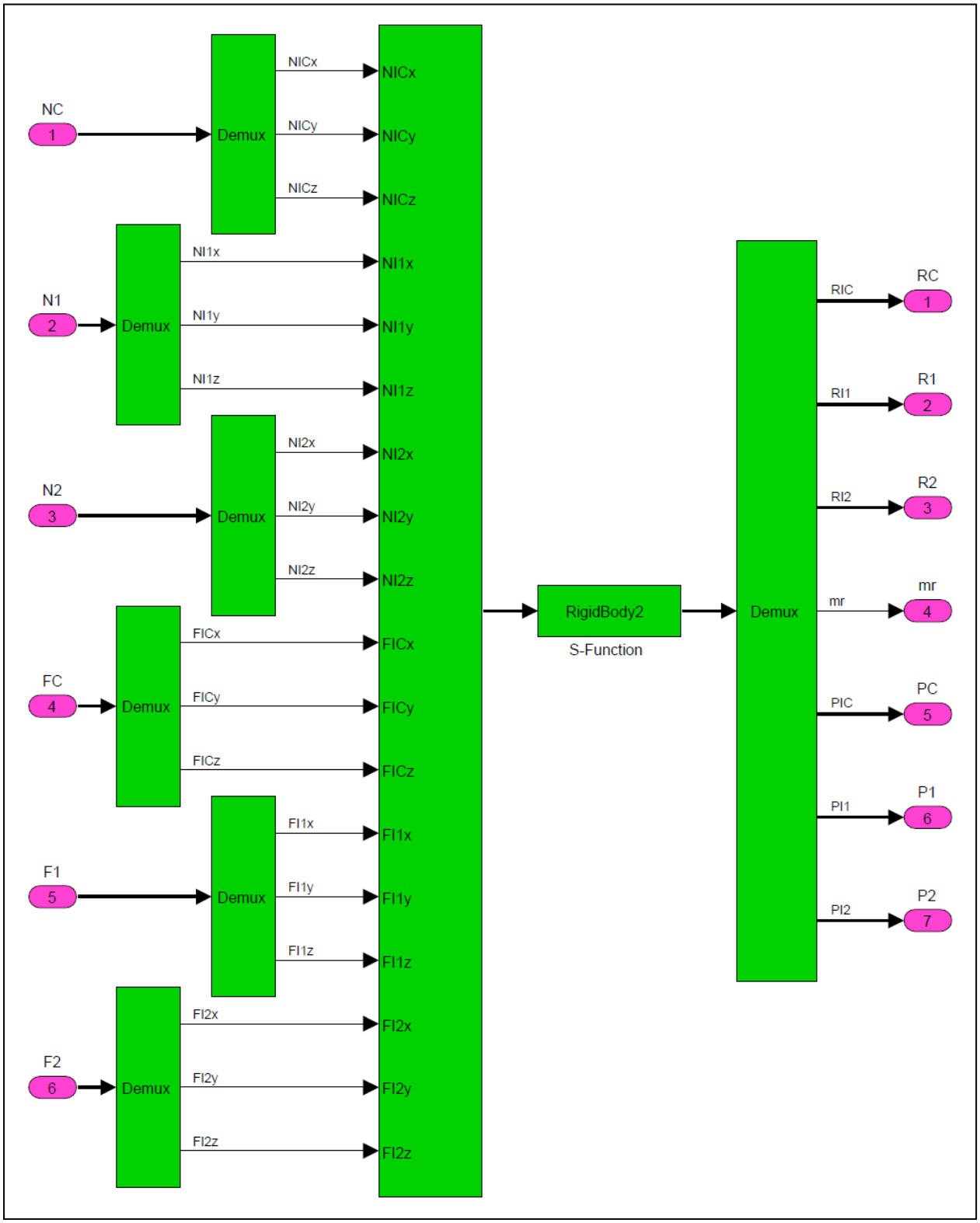


Figure 2. Structure of the RigidElement2 sub-system.

Appendix C
Smallbox

```
%*****************************************************************************************************************
% FILE    : Smallbox.m
%
% ABSTRACT:
% Script for calculating downwash flow field of Rooivalk by solving Talored Navier Stokes equations
%
% ORIGINATOR: Adrian Landman, May 2016.
% REVISION: A
%*****************************************************************************************************************
% Load the description of the Rooivalk rotor cone and throughwash as previously calculated with the Adoons.m program
clear;                                                                          % Vacate as much memory as possible
%*****************************************************************************************************************
% Prompt the user for parameters required to perform the calculation
%*****************************************************************************************************************
prompt   = {'Length of Field Box X-dimension in meter:',...
            'Length of Field Box Y-dimension in meter:',...
            'Length of Field Box Z-dimension in meter:',...
            'Size of spatial increment in meter:',...
            'Time duration of simulation',...
            'Size of time increment in seconds:'};
lines = 1;
def     = {'0.5','0.5','0.5','0.05','0.001','0.000001'};
answer   = inputdlg(prompt,'Please enter solver requirements',lines,def);

FBx=str2double(answer(1));                                          % Determine length of flight box along X-axis
FBy=str2double(answer(2));                                          % Determine length of flight box along Y-axis
FBz=str2double(answer(3));                                          % Determine length of flight box along Z-axis
delx=str2double(answer(4));                                                % Determine size of X-increment
dely=delx;                                                                 % Set Y-increment = X-increment
delz=delx;                                                                 % Set Z-increment = X-increment
Tsim=str2double(answer(5));                                              % Determine time span of simulation
delt=str2double(answer(6));                                               % Determine size of time increment

Ta=273+25;                                                  % Set ambient temperature to +25 degC for the moment
```

```
pa=1.013e+5;                                          % Set ambient pressure to pressure at STP conditions for the moment
WS=[0;0;0];                                                                % Set windspeed to zero for the moment

Ma=28.961e-3;                                                                % Molecular weight of air in kilogram
ua=1.861e-5;                                                              % Viscosity of air at +25 degC in N.s/m^2
Ca=1012;                                                     % Specific heat capacity of air at +25 degC in J/kg.degK
R=287;                                                          % Specific Gas Constant of air in J/kg.degK

%*************************************************************************************************************
% Set up the matrixes that will contain the flow field solution.  Initialize them all with zero so that a history of 3
% time steps is available.  Note that the amount of data is so large that we can only store the flow field at 4
% consecutive time increments in a FIFO configuration.  However, because of the large amount of data, we don't backshift
% the data after each time step because it would take too much time.  Rather, we overwrite the data of 4 time steps ago
% with the current data in the original position, using the FIFO index Fi to keep track of where we are.
%*************************************************************************************************************
Nx=fix(FBx/delx/2)*2+1;                                                    % Calculate number of nodes along the X-axis
Ny=fix(FBy/dely/2)*2+1;                                                    % Calculate number of nodes along the Y-axis
Nz=fix(FBz/delz/2)*2+1;                                                    % Calculate number of nodes along the Z-axis
p=ones(Nx,Ny,Nz,2);
for i=1:Nx                                % Initialize matrix p to reflect pressure distribution through field box at t=0
    x=((i-1)-(Nx-1)/2)*delx;
    for j=1:Ny
        for k=1:Nz
            p(i,j,k,1)=(0.9*(1-1/(1+exp(-10*x)))+0.1)*pa;
        end
    end
end
T=ones(Nx,Ny,Nz,2)*Ta;
u=ones(Nx,Ny,Nz,2)*0;
v=ones(Nx,Ny,Nz,2)*0;
w=ones(Nx,Ny,Nz,2)*0;

% Draw the initial condition
figure(1);
hold off;
cla;
colormap jet;

[X,Y,Z]=meshgrid(-FBx/2:delx:FBx/2, -FBy/2:dely:FBy/2, -FBz/2:delz:FBz/2);
```

```
data=p(:,:,:,1);
datas = smooth3(data,'gaussian',7);
isoval=0.0*pa;
h = patch(isosurface(X,Y,Z,datas,isoval),...
    'FaceColor','blue',...
    'EdgeColor','black',...
    'AmbientStrength',0.5,...
    'SpecularStrength',0.7,...
    'DiffuseStrength',0.4);
isonormals(datas,h);
patch(isocaps(X,Y,Z,datas,isoval),...
    'FaceColor','interp',...
    'EdgeColor','none');
colormap Jet;
daspect([1,1,1]);
%axis tight;
axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -FBz/2 +FBz/2]);
hi=(max(max(max(data))));
lo=(min(min(min(data))));
title('Air pressure (Pascal) in field box at t=0');
xlabel('X (meter)');
ylabel('Y (meter)');
zlabel('Z (meter)');
box on;
colorbar;
view(3);
drawnow;

% Determine the spatial sample matrixes and their inverses
TSMx=[delx^2, -delx, 1; 0, 0, 1; delx^2, delx, 1];
ITSMx=inv(TSMx);
TSMy=[dely^2, -dely, 1; 0, 0, 1; dely^2, dely, 1];
ITSMy=inv(TSMy);
TSMz=[delz^2, -delz, 1; 0, 0, 1; delz^2, delz, 1];
ITSMz=inv(TSMz);

% Prepare probes to hold sample data
Nt=fix(Tsim/delt);                                    % Calculate number of time steps in the simulation
Time=zeros(1,Nt);
```

```
Probep=zeros(1,Nt);
ProbeT=zeros(1,Nt);
Probeu=zeros(1,Nt);
Probev=zeros(1,Nt);
Probew=zeros(1,Nt);

% Prepare meshgrid to display 3-dimensional data
[XX,YY]=meshgrid(-FBx/2:delx:FBx/2, -FBy/2:dely:FBy/2);

% Start iteration with page 1
frame=0;
Ip=2;                                          % Set page index to two to ensure first calculation cycle uses page 1
for cycle=1:Nt
    disp(['PASS1: cycle ',num2str(cycle),' of ',num2str(Nt)]); % Debug code
    disp(['-----------']);
    pause(0.001);

    if (Ip==1)                                 % Toggle pointers to the two pages of the result matrixes
        Ip=2;                                  % Point to page1 as base values
        Ipo=1;                                 % Set opposite pointer to page2
    else
        Ip=1;                                  % Point to page2 as base values
        Ipo=2;                                 % Set opposite pointer to page1
    end

    % PASS1: first calculate new state of all nodes except those at boundaries of field box.
    for i=1:Nx
        x=((i-1)-(Nx-1)/2)*delx;                           % Calculate current x-position of test volume
        for j=1:Ny
            y=((j-1)-(Ny-1)/2)*dely;                       % Calculate current y-position of test volume
            for k=1:Nz
                z=((k-1)-(Nz-1)/2)*delz;                   % Calculate current z-position of test volume

                % Perform iteration calcs if current test volume is not on edge of field box
                if not((i==1)||(i==Nx)||(j==1)||(j==Ny)||(k==1)||(k==Nz))
                    A11=(u(i,j,k,Ip)^2+v(i,j,k,Ip)^2+w(i,j,k,Ip)^2)/(2*R*T(i,j,k,Ip))+Ca/R;
                    A12=-3*p(i,j,k,Ip)/(2*R*T(i,j,k,Ip)^2);
                    A13=u(i,j,k,Ip)*p(i,j,k,Ip)/(R*T(i,j,k,Ip));
                    A14=v(i,j,k,Ip)*p(i,j,k,Ip)/(R*T(i,j,k,Ip));
```

```
A15=w(i,j,k,Ip)*p(i,j,k,Ip)/(R*T(i,j,k,Ip));
A21=0;
A22=0;
A23=p(i,j,k,Ip)/(R*T(i,j,k,Ip));
A24=0;
A25=0;
A31=0;
A32=0;
A33=0;
A34=p(i,j,k,Ip)/(R*T(i,j,k,Ip));
A35=0;
A41=0;
A42=0;
A43=0;
A44=0;
A45=p(i,j,k,Ip)/(R*T(i,j,k,Ip));
A51=1/T(i,j,k,Ip);
A52=-p(i,j,k,Ip)/T(i,j,k,Ip)^2;
A53=0;
A54=0;
A55=0;
A=[A11,A12,A13,A14,A15;...
   A21,A22,A23,A24,A25;...
   A31,A32,A33,A34,A35;...
   A41,A42,A43,A44,A45;...
   A51,A52,A53,A54,A55];

dpx=ITSMx*[p(i-1,j,k,Ip);p(i,j,k,Ip);p(i+1,j,k,Ip)];      % Calculate derivatives for p along X-axis
dpy=ITSMy*[p(i,j-1,k,Ip);p(i,j,k,Ip);p(i,j+1,k,Ip)];      % Calculate derivatives for p along Y-axis
dpz=ITSMz*[p(i,j,k-1,Ip);p(i,j,k,Ip);p(i,j,k+1,Ip)];      % Calculate derivatives for p along Z-axis
dTx=ITSMx*[T(i-1,j,k,Ip);T(i,j,k,Ip);T(i+1,j,k,Ip)];      % Calculate derivatives for T along X-axis
dTy=ITSMy*[T(i,j-1,k,Ip);T(i,j,k,Ip);T(i,j+1,k,Ip)];      % Calculate derivatives for T along Y-axis
dTz=ITSMz*[T(i,j,k-1,Ip);T(i,j,k,Ip);T(i,j,k+1,Ip)];      % Calculate derivatives for T along Z-axis
dux=ITSMx*[u(i-1,j,k,Ip);u(i,j,k,Ip);u(i+1,j,k,Ip)];      % Calculate derivatives for u along X-axis
duy=ITSMy*[u(i,j-1,k,Ip);u(i,j,k,Ip);u(i,j+1,k,Ip)];      % Calculate derivatives for u along Y-axis
duz=ITSMz*[u(i,j,k-1,Ip);u(i,j,k,Ip);u(i,j,k+1,Ip)];      % Calculate derivatives for u along Z-axis
dvx=ITSMx*[v(i-1,j,k,Ip);v(i,j,k,Ip);v(i+1,j,k,Ip)];      % Calculate derivatives for v along X-axis
dvy=ITSMy*[v(i,j-1,k,Ip);v(i,j,k,Ip);v(i,j+1,k,Ip)];      % Calculate derivatives for v along Y-axis
dvz=ITSMz*[v(i,j,k-1,Ip);v(i,j,k,Ip);v(i,j,k+1,Ip)];      % Calculate derivatives for v along Z-axis
```

```
                dwx=ITSMx*[w(i-1,j,k,Ip);w(i,j,k,Ip);w(i+1,j,k,Ip)];      % Calculate derivatives for w along X-axis
                dwy=ITSMy*[w(i,j-1,k,Ip);w(i,j,k,Ip);w(i,j+1,k,Ip)];      % Calculate derivatives for w along Y-axis
                dwz=ITSMz*[w(i,j,k-1,Ip);w(i,j,k,Ip);w(i,j,k+1,Ip)];      % Calculate derivatives for w along Z-axis
                B1=-ua*u(i,j,k,Ip)*(2*duz(1)+2*duy(1))+...
                   -ua*v(i,j,k,Ip)*(2*dvz(1)+2*dvx(1))+...
                   -ua*w(i,j,k,Ip)*(2*dwx(1)+2*dwy(1))+...
                   -(Ca/R+1)*(w(i,j,k,Ip)*dpz(2)+u(i,j,k,Ip)*dpx(2)+v(i,j,k,Ip)*dpy(2))+...
                   -p(i,j,k,Ip)*(Ca/R+1)*(dwz(2)+dux(2)+dvy(2));
                B2=-dpx(2);
                B3=-dpy(2);
                B4=-dpz(2);
                B5=-p(i,j,k,Ip)/T(i,j,k,Ip)*(dwz(2)+dux(2)+dvy(2))+...
                   -1/T(i,j,k,Ip)*(w(i,j,k,Ip)*dpz(2)+u(i,j,k,Ip)*dpx(2)+v(i,j,k,Ip)*dpy(2))+...
                    p(i,j,k,Ip)/T(i,j,k,Ip)^2*(w(i,j,k,Ip)*dTz(2)+u(i,j,k,Ip)*dTx(2)+v(i,j,k,Ip)*dTy(2));
                B=[B1;B2;B3;B4;B5];

                DA=inv(A)*B;            % Calculate partial time derivatives of p, T, u, v and w at current position
                p(i,j,k,Ipo)=p(i,j,k,Ip)+DA(1)*delt;                       % Save evoluted pressure in opposite page
                T(i,j,k,Ipo)=T(i,j,k,Ip)+DA(2)*delt;                   % Save evoluted temperature in opposite page
                u(i,j,k,Ipo)=u(i,j,k,Ip)+DA(3)*delt;      % Save evoluted air velocity along X-axis in opposite page
                v(i,j,k,Ipo)=v(i,j,k,Ip)+DA(4)*delt;      % Save evoluted air velocity along Y-axis in opposite page
                w(i,j,k,Ipo)=w(i,j,k,Ip)+DA(5)*delt;      % Save evoluted air velocity along Z-axis in opposite page

                % Save parameter values if test volume is at field box center
                if ((i==(Nx-1)/2+1)&&(j==(Ny-1)/2+1)&&(k==(Nz-1)/2+1))
                    Time(cycle)=cycle*delt;
                    Probep(cycle)=p(i,j,k,Ipo);
                    ProbeT(cycle)=T(i,j,k,Ipo);
                    Probeu(cycle)=u(i,j,k,Ipo);
                    Probev(cycle)=v(i,j,k,Ipo);
                    Probew(cycle)=w(i,j,k,Ipo);
                end
            end
        end
    end
end

% PASS2: now set states of boundary cubes
for i=1:Nx
```

```
for j=1:Ny
    for k=1:Nz
        if (i==1)
            p(i,j,k,Ipo)=p(i+1,j,k,Ipo);
            T(i,j,k,Ipo)=T(i+1,j,k,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        elseif (i==Nx)
            p(i,j,k,Ipo)=p(i-1,j,k,Ipo);
            T(i,j,k,Ipo)=T(i-1,j,k,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        elseif (j==1)
            p(i,j,k,Ipo)=p(i,j+1,k,Ipo);
            T(i,j,k,Ipo)=T(i,j+1,k,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        elseif (j==Ny)
            p(i,j,k,Ipo)=p(i,j-1,k,Ipo);
            T(i,j,k,Ipo)=T(i,j-1,k,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        elseif (k==1)
            p(i,j,k,Ipo)=p(i,j,k+1,Ipo);
            T(i,j,k,Ipo)=T(i,j,k+1,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        elseif (k==Nz)
            p(i,j,k,Ipo)=p(i,j,k-1,Ipo);
            T(i,j,k,Ipo)=T(i,j,k-1,Ipo);
            u(i,j,k,Ipo)=0;
            v(i,j,k,Ipo)=0;
            w(i,j,k,Ipo)=0;
        end
    end
```

```
    end
end

% Draw the results of the simulation at multiples of 50 cycles
if ((cycle==1)||(round(cycle/50)*50-cycle==0))
    t=cycle*delt;
    frame=frame+1;
    figure(2);
    data=u(:,:,:,1);
    datas = smooth3(data,'gaussian',7);
    subplot(2,3,1);
    isoval=50;
    h = patch(isosurface(X,Y,Z,datas,isoval),...
        'FaceColor','blue',...
        'EdgeColor','black',...
        'AmbientStrength',0.5,...
        'SpecularStrength',0.7,...
        'DiffuseStrength',0.4);
    isonormals(datas,h);
    patch(isocaps(X,Y,Z,datas,isoval),...
        'FaceColor','interp',...
        'EdgeColor','none');
    daspect([1,1,1]);
    axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -FBz/2 +FBz/2]);
    %axis tight;
    title(['u above 50m/s at t=',num2str(t)]);
    xlabel('X (meter)');
    ylabel('Y (meter)');
    zlabel('Z (meter)');
    box on;
    colorbar;
    view(3);
    subplot(2,3,2);
    dummy=p(:,:,6,1);
    %B = reshape(dummy,Nx,Ny);
    surfl(XX,YY,dummy);
    view(3);
    axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 0 1.7e+5]);
    title(['Air pressure p over XY plane at Z=0']);
```

```
xlabel('X');
ylabel('Y');
zlabel('Pressure (Pascal)');
subplot(2,3,3);
dummy=T(:,:,6,1);
%B = reshape(dummy,Nx,Ny);
surfl(XX,YY,dummy);
view(3);
axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 0 700]);
title(['Air temperature T over XY plane at Z=0']);
xlabel('X');
ylabel('Y');
zlabel('Temperature (°K)');
subplot(2,3,4);
dummy=u(:,:,6,1);
%B = reshape(dummy,Nx,Ny);
surfl(XX,YY,dummy);
view(3);
axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -310 310]);
title(['Air speed u over XY plane at Z=0']);
xlabel('X');
ylabel('Y');
zlabel('Speed (m/s)');
subplot(2,3,5);
dummy=v(:,:,6,1);
%B = reshape(dummy,Nx,Ny);
surfl(XX,YY,dummy);
view(3);
axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -0.003 0.003]);
title(['Air speed v over XY plane at Z=0']);
xlabel('X');
ylabel('Y');
zlabel('Speed (m/s)');
subplot(2,3,6);
dummy=w(:,:,6,1);
%B = reshape(dummy,Nx,Ny);
surfl(XX,YY,dummy);
view(3);
axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -0.001 0.001]);
title(['Air speed w over XY plane at Z=0']);
```

```
xlabel('X');
ylabel('Y');
zlabel('Speed (m/s)');

% Plot the variation of parameters of air in center test volume with time
figure(3);
hold off;
cla;
subplot(2,3,1);
plot(Time(1:cycle),Probep(1:cycle),'LineWidth',2,'Color',[0 0 1]);
%axis([0 5e-3 0 1.5e+5]);
title('Air pressure');
xlabel('Time (sec)');
ylabel('Pressure (Pascal)');
box on;
grid on;
subplot(2,3,2);
plot(Time(1:cycle),ProbeT(1:cycle),'LineWidth',2,'Color',[0 0 1]);
%axis([0 5e-3 200 400]);
title('Air temperature');
xlabel('Time (sec)');
ylabel('Temperature (°K)');
box on;
grid on;
subplot(2,3,3);
plot(Time(1:cycle),Probeu(1:cycle),'LineWidth',2,'Color',[0 0 1]);
%axis([0 5e-3 -300 300]);
title('Speed u');
xlabel('Time (sec)');
ylabel('Speed (m/s)');
box on;
grid on;
subplot(2,3,4);
plot(Time(1:cycle),Probev(1:cycle),'LineWidth',2,'Color',[0 0 1]);
%axis([0 5e-3 -1e-3 1e-3]);
title('Speed v');
xlabel('Time (sec)');
ylabel('Speed (m/s)');
box on;
```

```
        grid on;
        subplot(2,3,5);
        plot(Time(1:cycle),Probew(1:cycle),'LineWidth',2,'Color',[0 0 1]);
        %axis([0 5e-3 -1e-3 1e-3]);
        title('Speed w');
        xlabel('Time (sec)');
        ylabel('Speed (m/s)');
        box on;
        grid on;

        figure(3);
        hold off;
        cla;
        dummy=v(:,:,6,1);
        surfl(XX,YY,dummy);
        view(3);
        %axis([-FBx/2 +FBx/2 -FBy/2 +FBy/2 -0.005 0.005]);
        title(['Air speed v over XY plane at Z=0, t=',num2str(t)]);
        xlabel('X');
        ylabel('y');
        zlabel('Speed (m/s)');
        F(frame)=getframe;

        drawnow;
    end
end
figure(3);
hold off;
cla;
movie(F,5);                                                                    % Play the sequence
```